

Introduction to MemComputing and its Applications in High Performance Computing

Fabio L. Traversa, Ph.D., CTO



MemComputing Inc.

approved for public release

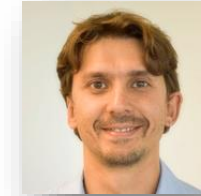


Spun out 2016



Founders:

- Dr. Fabio Traversa, Co-Inventor, CTO
- Dr. Max Di Ventra, Co-Inventor
- John Beane, CEO, Serial Entrepreneur



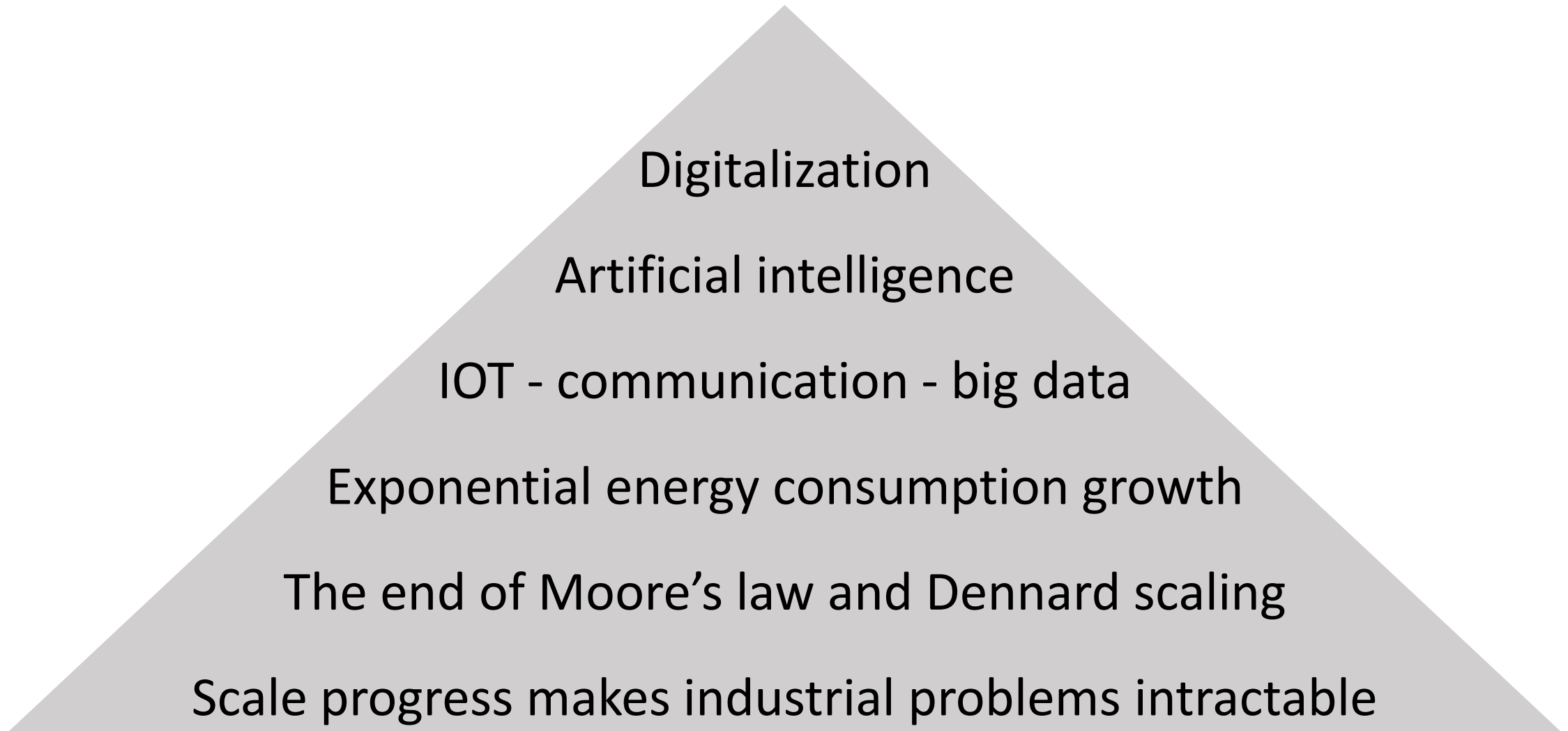
Mission

Develop innovative computing platforms based on patented
Self-organizing Circuits and Computational Memories

Purpose

Overcome computational limits industry faces today and
tomorrow

Pyramid of Motivations

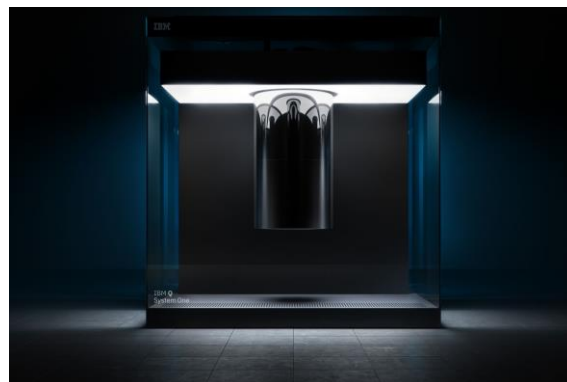


The solution:
Non-von
Neumann
architectures



MemComputing

**Neuromorphic
computing**

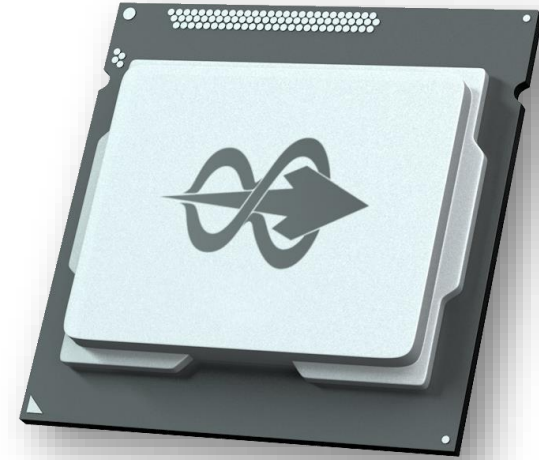


**Quantum
computing**

Products

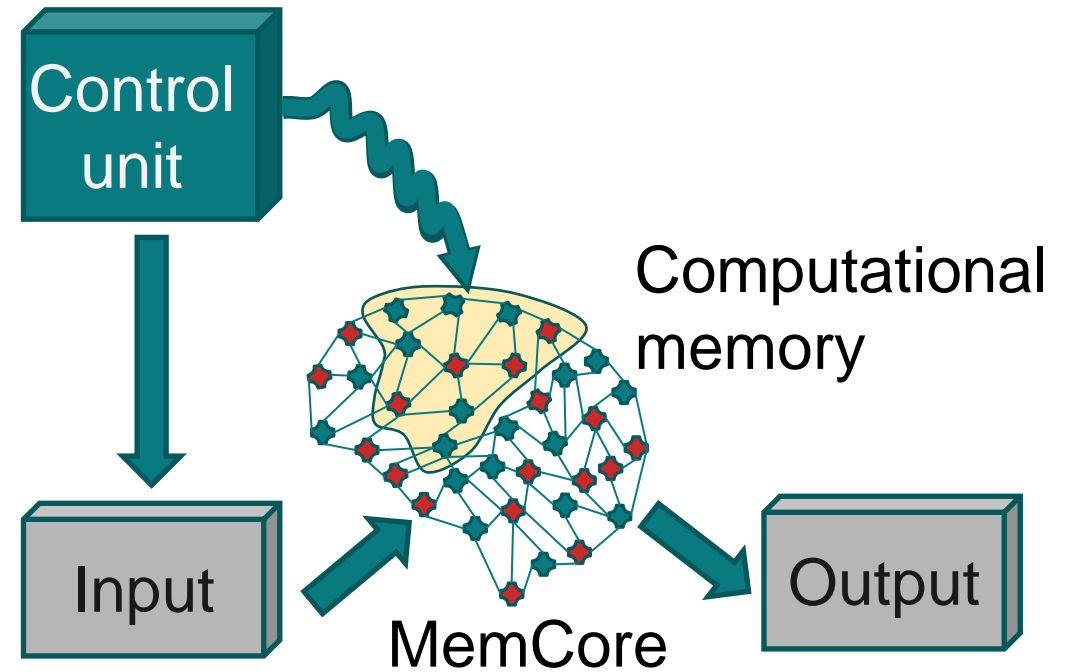


Optimization Problems

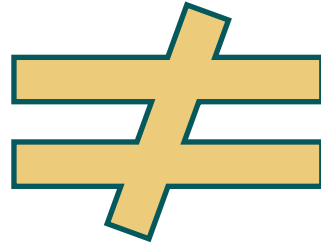
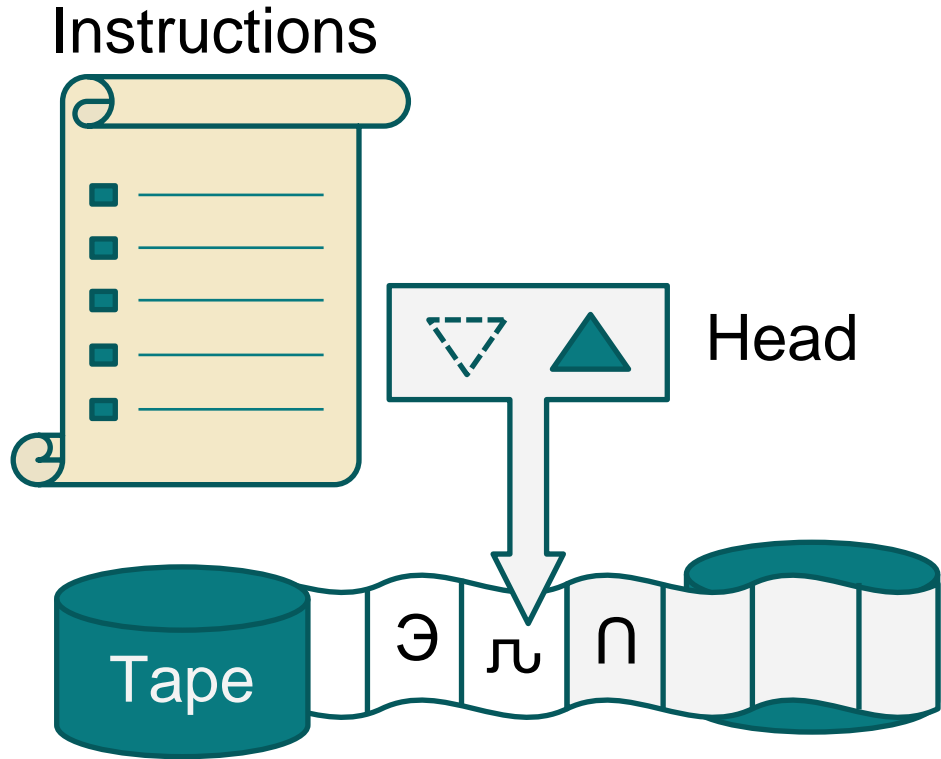


Real time computing
(AI & NN, graphics, edge, comms)

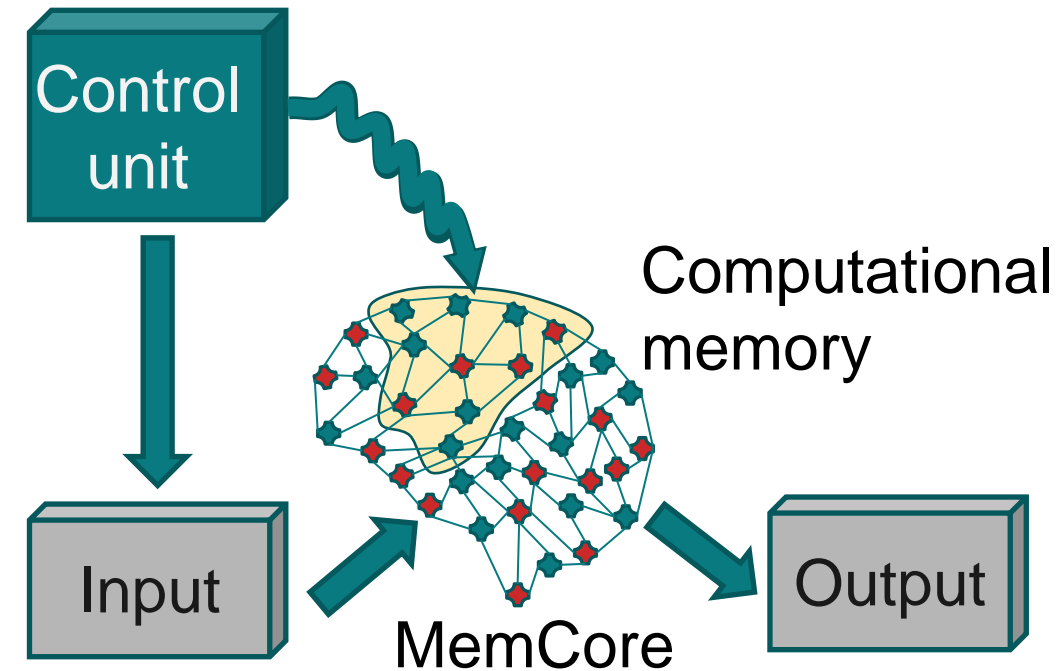
Universal Memcomputing Machine



Turing



Memcomputing



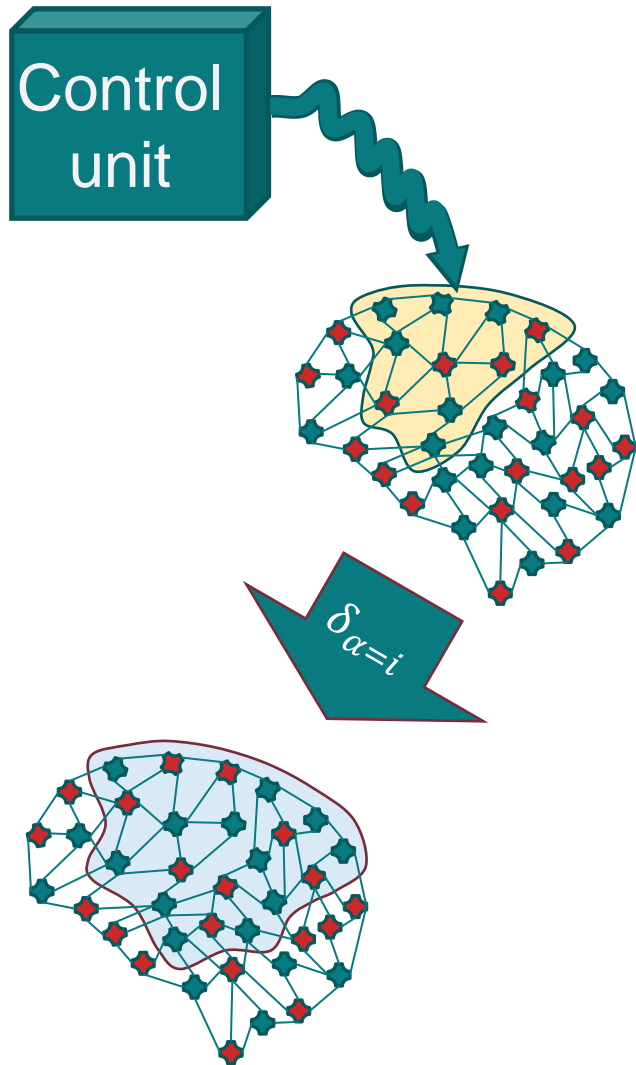
- Sequential
- General purpose (algorithm adapts problem to the machine)

- Intrinsically parallel
- Adaptive (Machine adapts to the problem)

Direct implications

- **Mitigate or eliminate entirely the von Neumann bottleneck**
- **Ultra low power and extreme performance distributed computing architectures**
- **Efficient solution of Turing (combinatorial) complex problems**

Computational Complexity Benefit



formal
definition

$$UMM = (M, \Delta, \mathcal{P}, S, \Sigma, p_0, s_0, F)$$
$$\delta_{\alpha} : M^{m_{\alpha}} \setminus F \times \mathcal{P} \rightarrow M^{m'_{\alpha}} \times \mathcal{P}^2 \times S$$



Equivalent to Non-deterministic Turing Machine

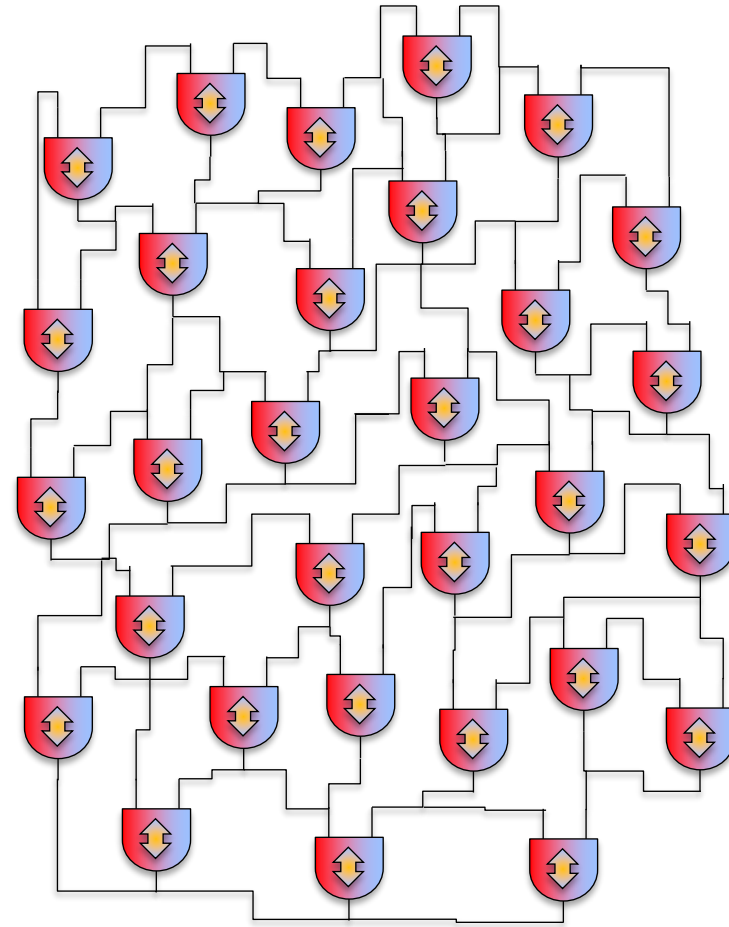


Efficient solution of NP problems
within the Memcomputing Paradigm

The Challenge

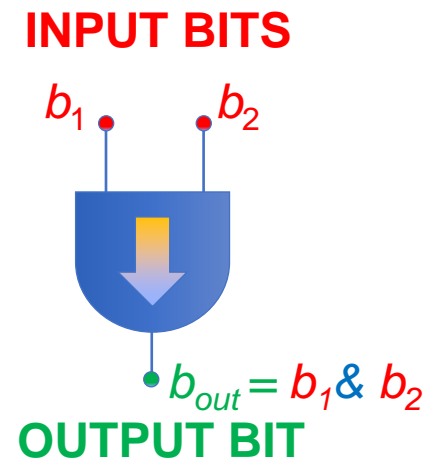
*Design a practical MemComputing
Machine*

Self-organizing circuits



Boolean Logic

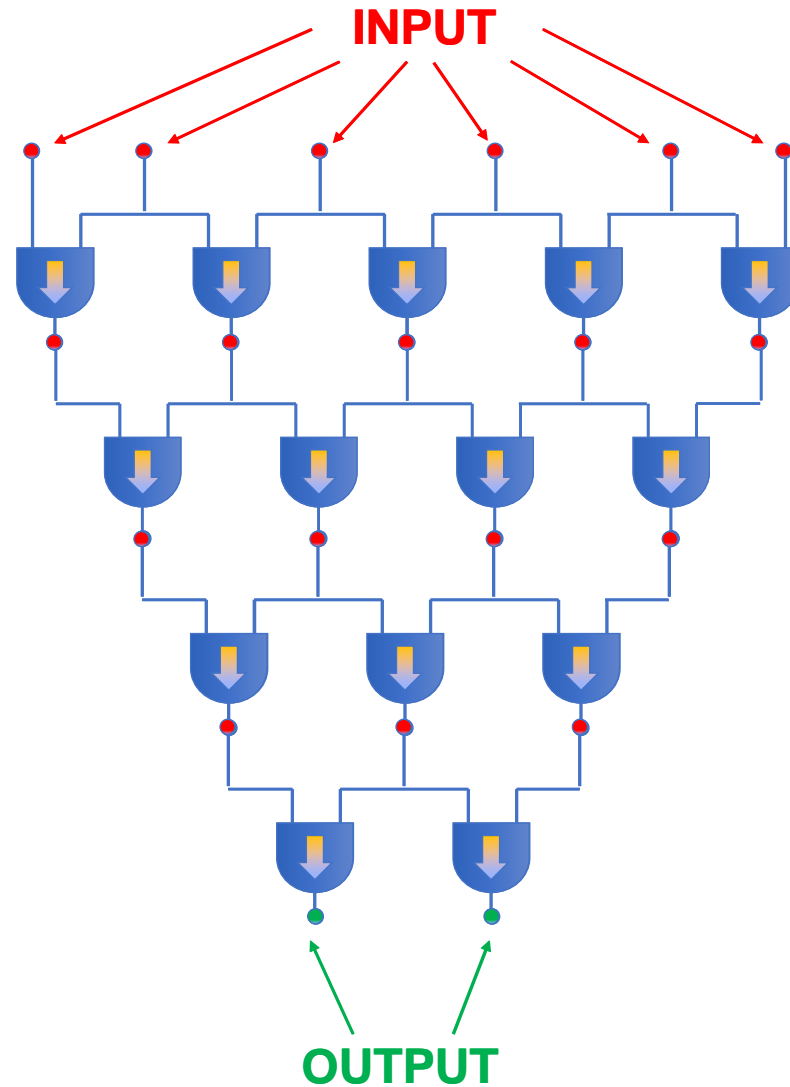
Conventional logic gate



b_1	b_2	b_{out}
1	1	1
1	0	0
0	1	0
0	0	0

F.L. Traversa and M. Di Ventra,
UCSD Patent (2015), Chaos (2017)

Boolean Logic

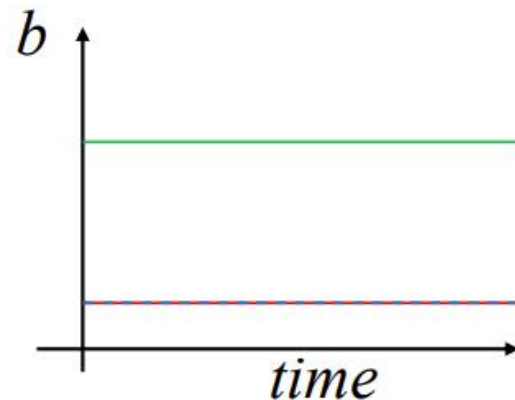


F.L. Traversa and M. Di Ventra,
UCSD Patent (2015), Chaos (2017)

Self-organizing Logic

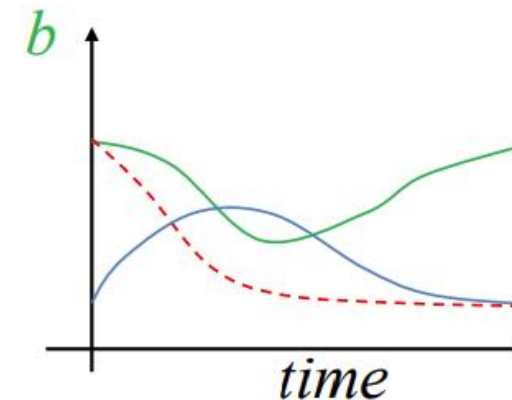
Logic relation
satisfied \Leftrightarrow Stable
signal configuration

$$b_o = b_1 \& b_2$$



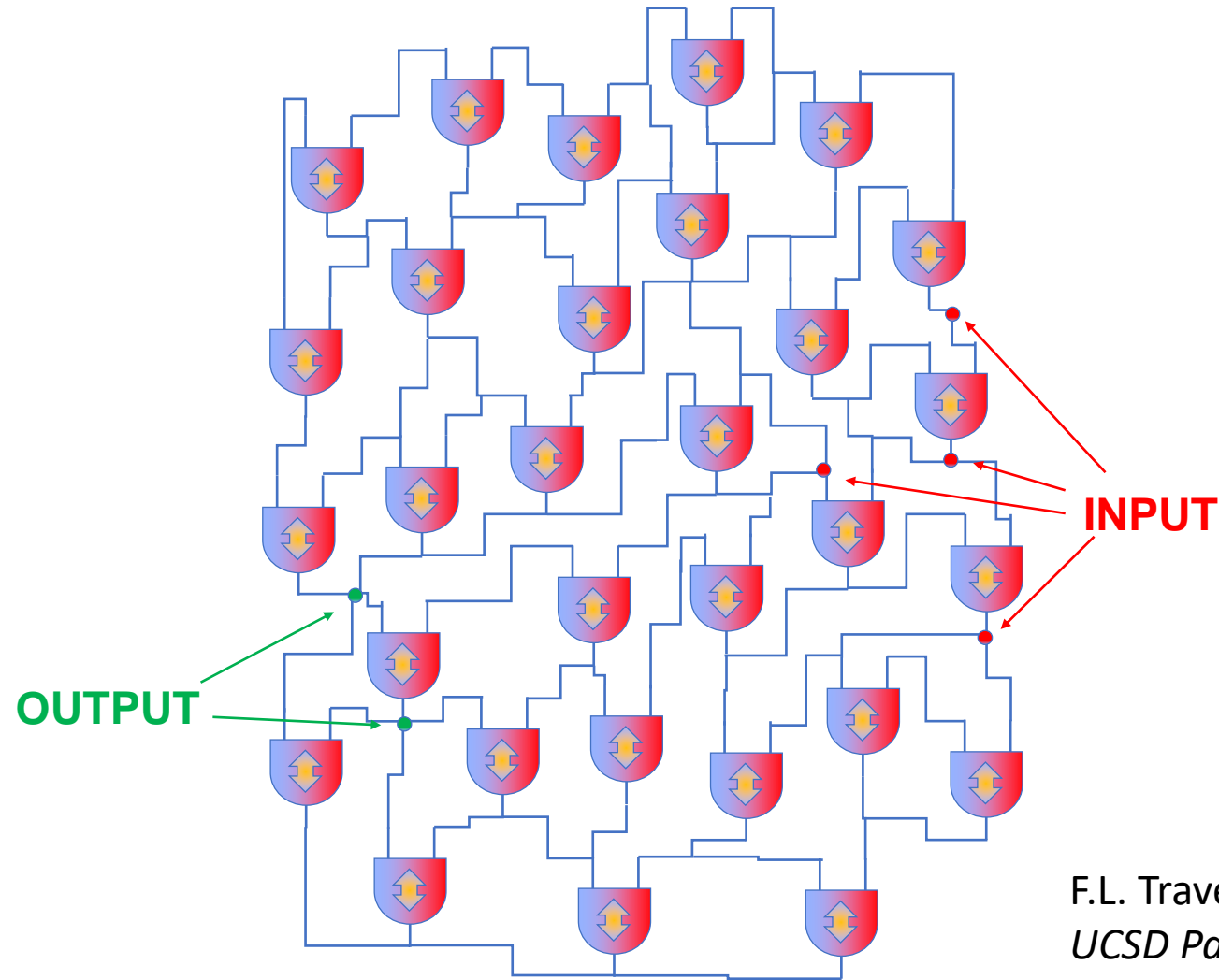
Logic relation not
satisfied \Leftrightarrow Unstable
signal configuration

$$b_o \neq b_1 \& b_2$$



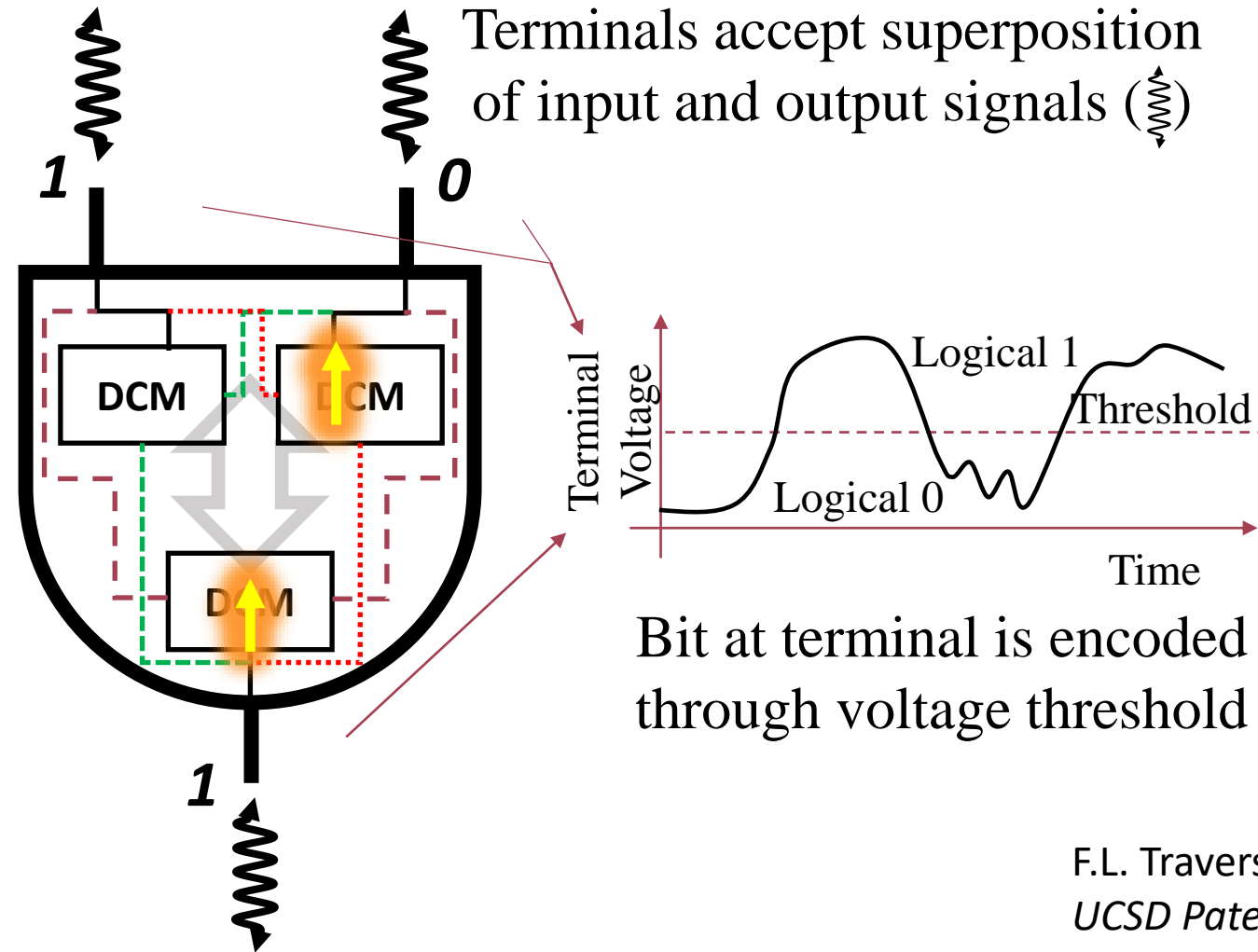
F.L. Traversa and M. Di Ventra,
UCSD Patent (2015), Chaos (2017)

Self-organizing Logic



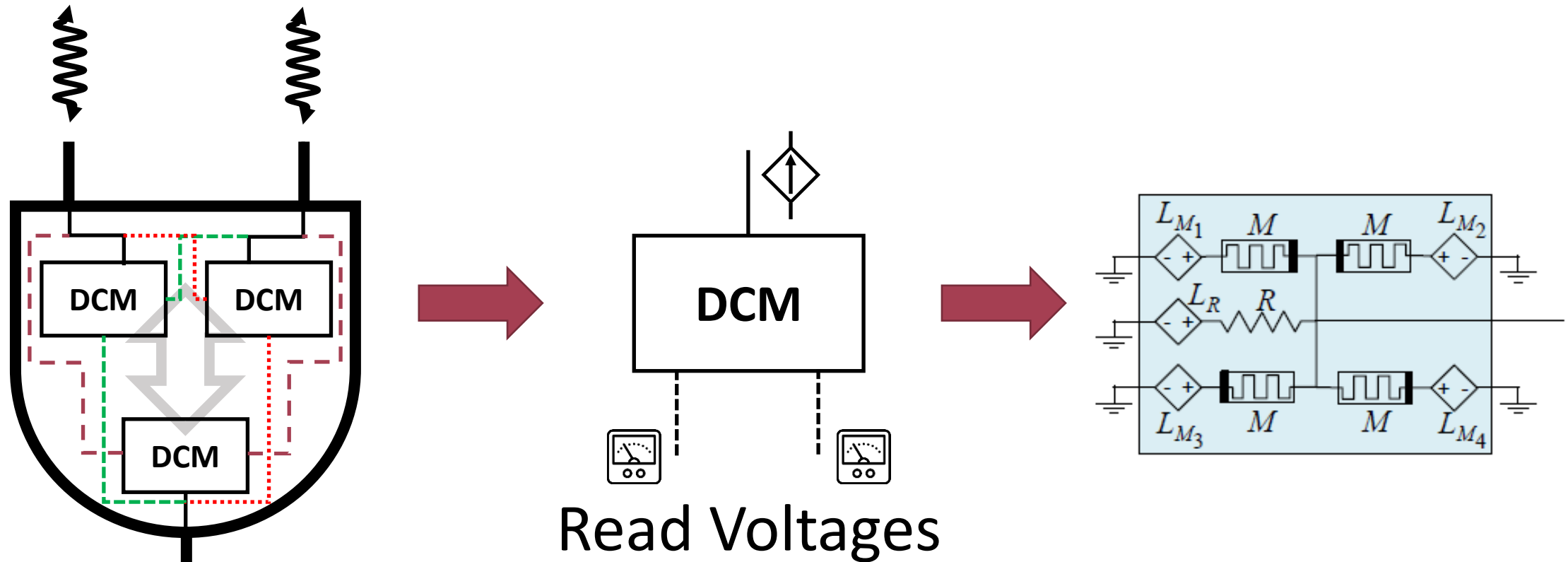
F.L. Traversa and M. Di Ventra,
UCSD Patent (2015), Chaos (2017)

Self-organizing logic gates



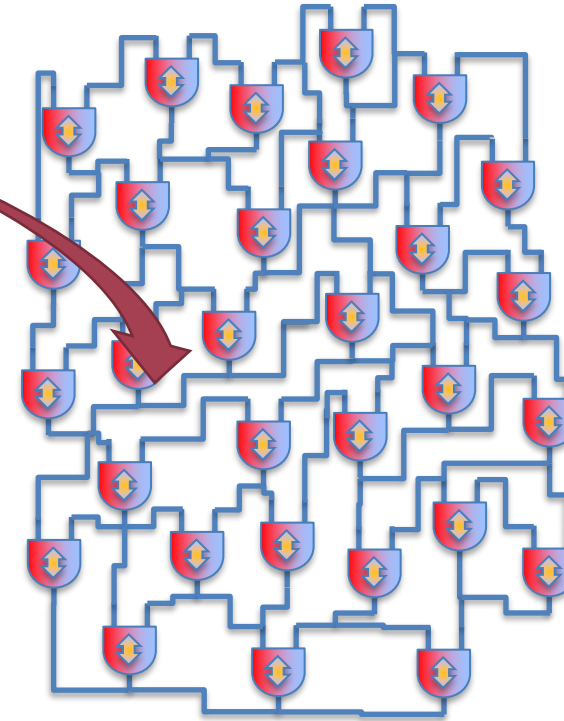
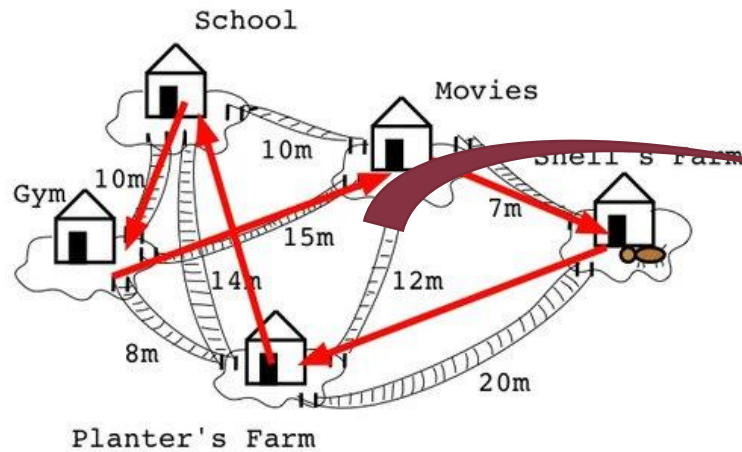
F.L. Traversa and M. Di Ventra,
UCSD Patent (2015), Chaos (2017)

Self-organizing logic gates



F.L. Traversa and M. Di Ventra,
UCSD Patent (2015), Chaos (2017)

From problem to solution

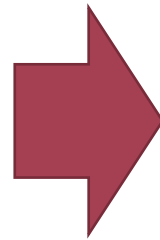


$$\begin{aligned}
 & (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6) \wedge \\
 & (\bar{x}_1 \vee (x_2 \vee x_3 \vee x_4 \vee x_5 \vee x_6)) \wedge \\
 & (\bar{x}_2 \vee (x_1 \vee x_3 \vee x_4 \vee x_5 \vee x_6)) \wedge \\
 & (\bar{x}_3 \vee (x_1 \vee x_2 \vee x_4 \vee x_5 \vee x_6)) \wedge \\
 & (\bar{x}_4 \vee (x_1 \vee x_2 \vee x_3 \vee x_5 \vee x_6)) \wedge \\
 & (\bar{x}_5 \vee (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_6)) \wedge \\
 & (\bar{x}_6 \vee (x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5))
 \end{aligned}$$

F.L. Traversa and M. Di Ventra,
Chaos (2017)

Self-Organizing Circuit Design Principles

- Functional analysis
- Topology and Topological field theory
- Stability Theory
- Chaos Theory



- Attractors and equilibria
- Convergence properties
- Control
- Absence of Chaos
- Criticality

Formal proofs:

F.L. Traversa and M. Di Ventra, *Chaos* (2017);
M. Di Ventra and F.L. Traversa, *Phys. Lett. A* (2017);
M. Di Ventra and F.L. Traversa, *Chaos* (2017)

Further readings:

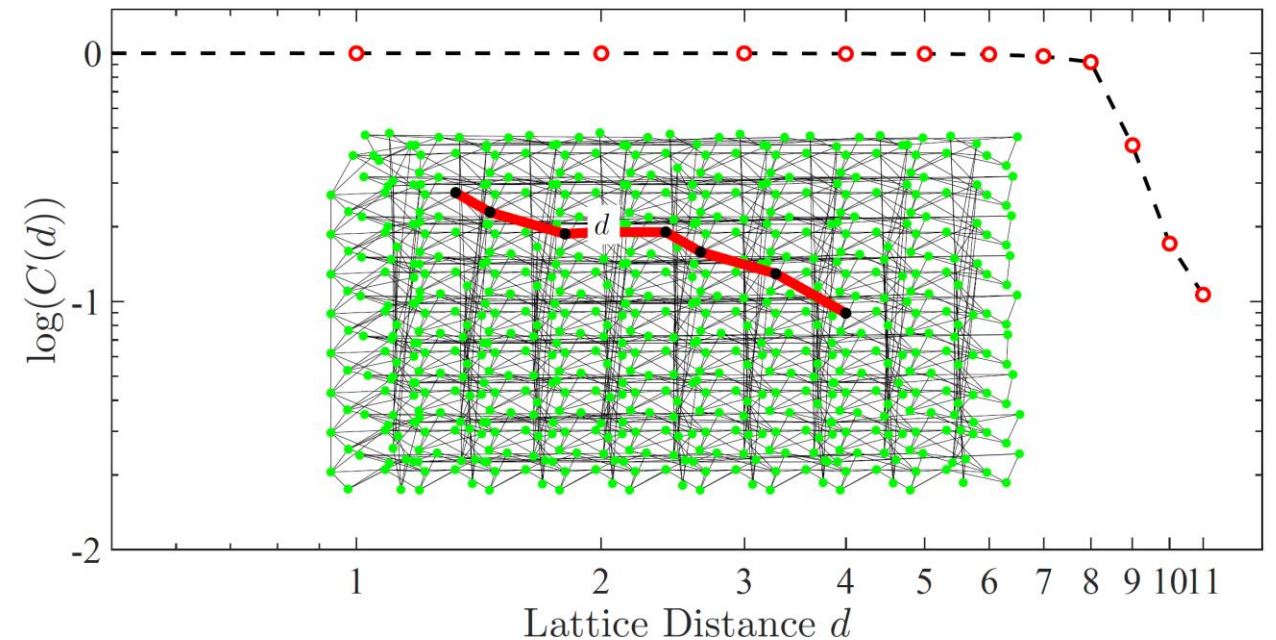
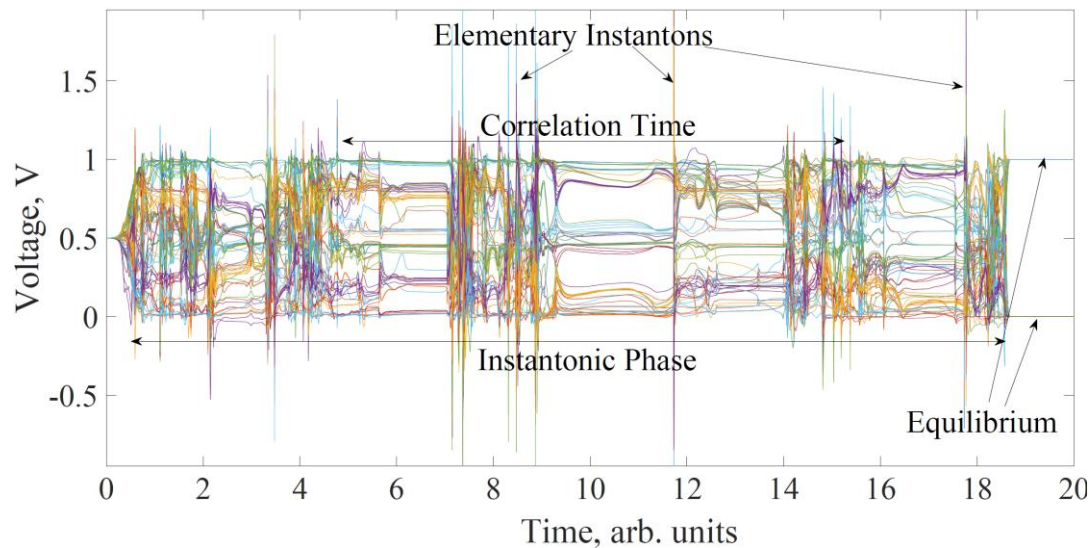
F. Caravelli F.L. Traversa and M. Di Ventra, *Phys Rev E* (2017);
F. Caravelli, *Entropy* (2018);
S. Bearden, F. Sheldon, M. Di Ventra, *EPL* (2019)

Self-Organization & Non-Locality

- System is critical (edge of chaos)
- System has equilibria
- System is point dissipative

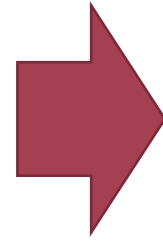


- Scale-free correlations
- Optimal convergence

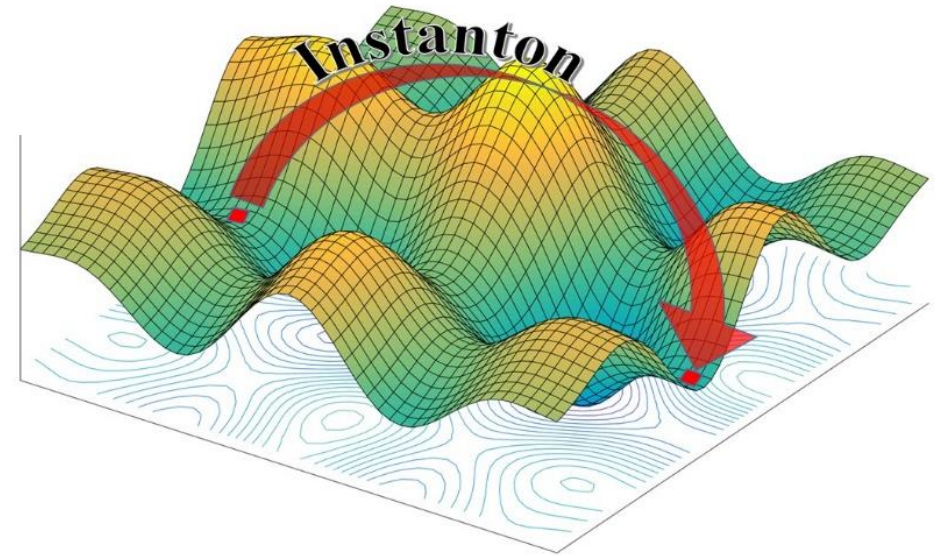
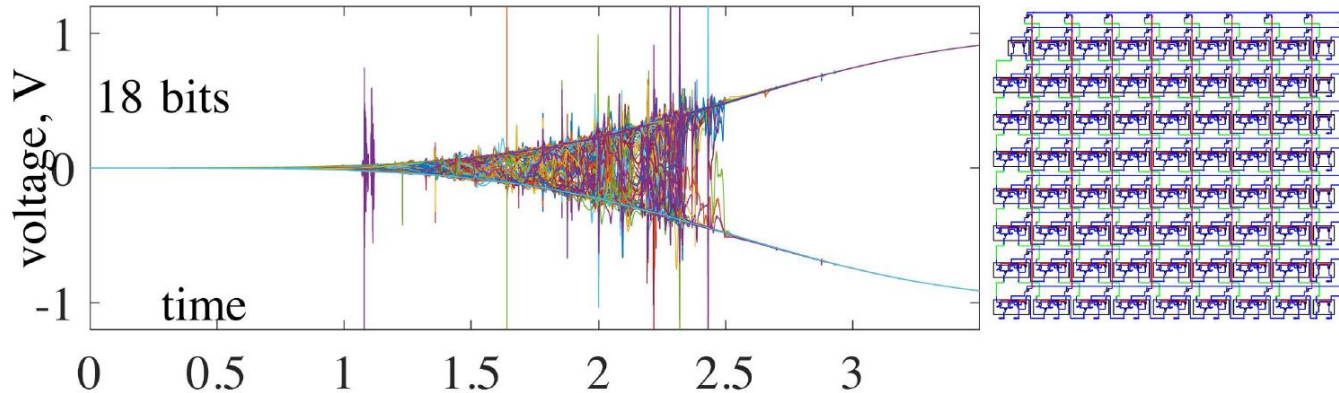


Instantonic Phase and Tunneling

- Scale free correlations
- Multidimensional state space
- “Hidden” state variables



- Instantonic Phase
- Convexification
- Classical Tunneling

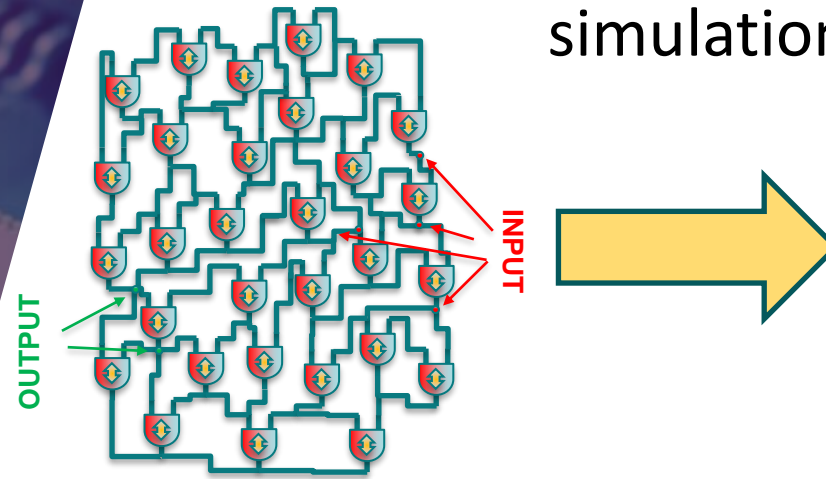


F. Caravelli, F. Sheldon, F.L. Traversa
(Arxiv 2021)

M. Di Ventra, F.L. Traversa, I.V. Ovchinnikov,
(Annalen der Physik 2017)

Proprietary

Virtual MemComputing Machine

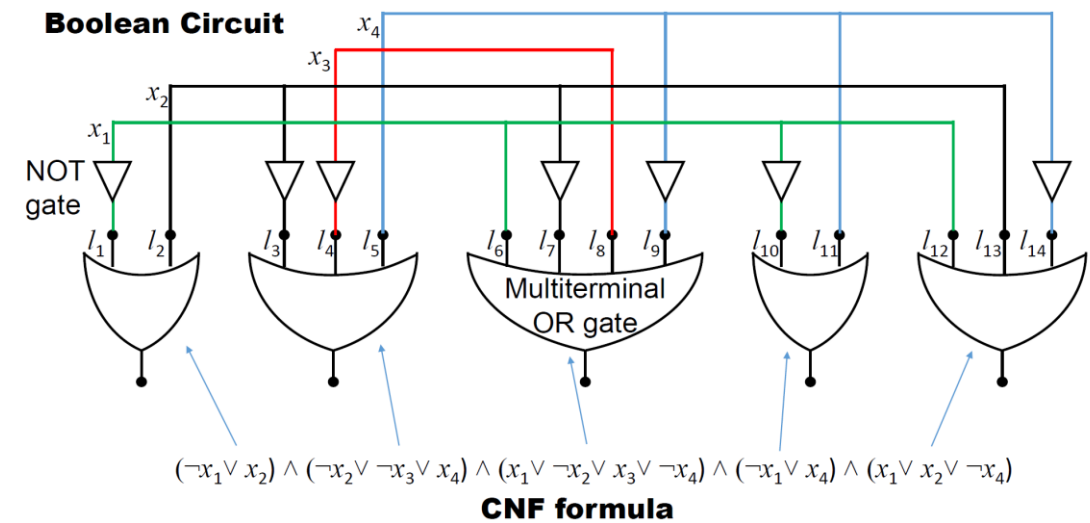


Software
simulation



Emulation of self-organizing circuits enables a radically different and more efficient use of the standard hardware to solve Combinatorial Optimization Problems

Solving Satisfiability problems



Maximum Satisfiability Problem

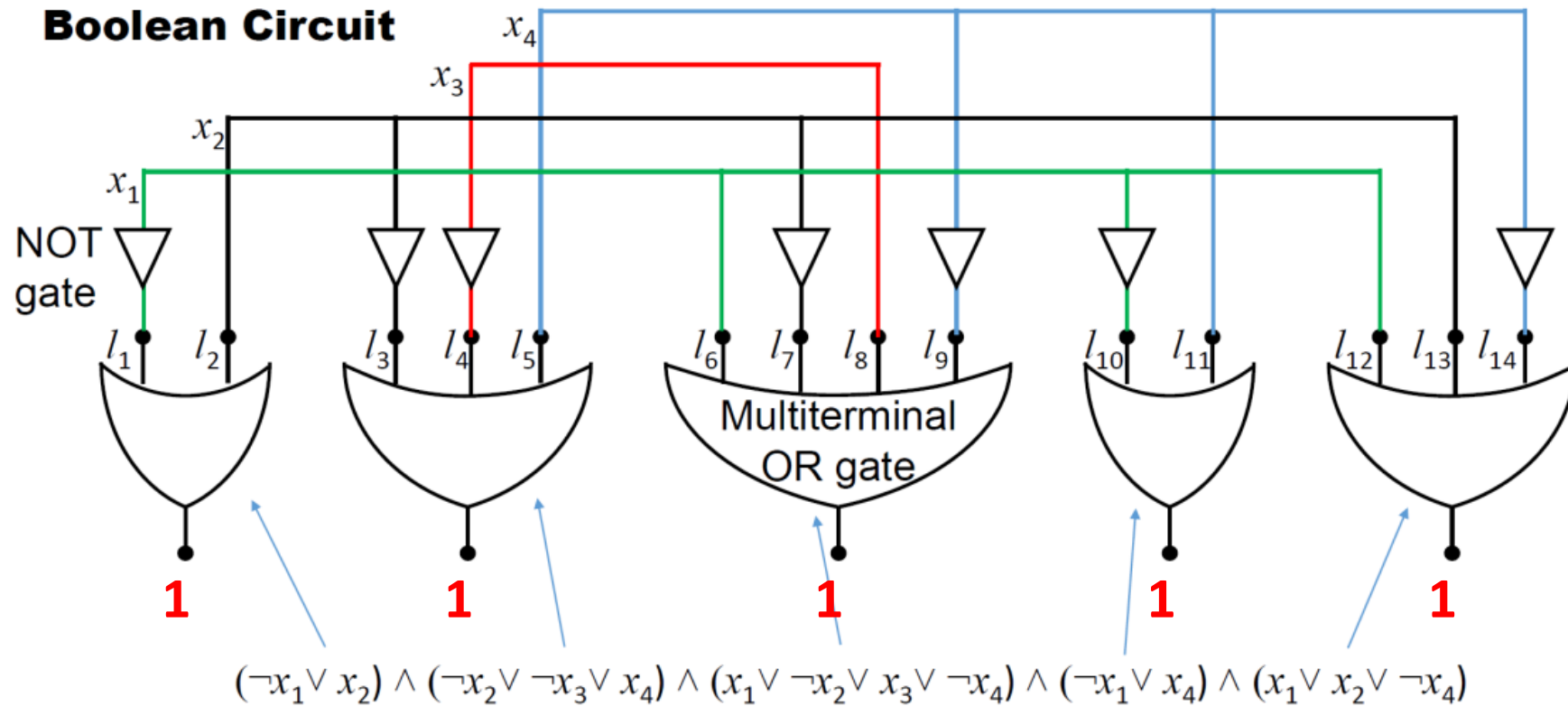
$$f = \underbrace{(x_1 \vee x_2 \vee \neg x_3)}_{\text{clause}} \wedge \underbrace{(x_3 \vee \neg x_4)}_{\text{clause}} \wedge \dots \wedge (\neg x_k \vee x_n)$$

Literal *OR* *AND*

Goal:

Maximize the number of satisfied clauses
or, equivalently,
minimize the number of unsatisfied clauses

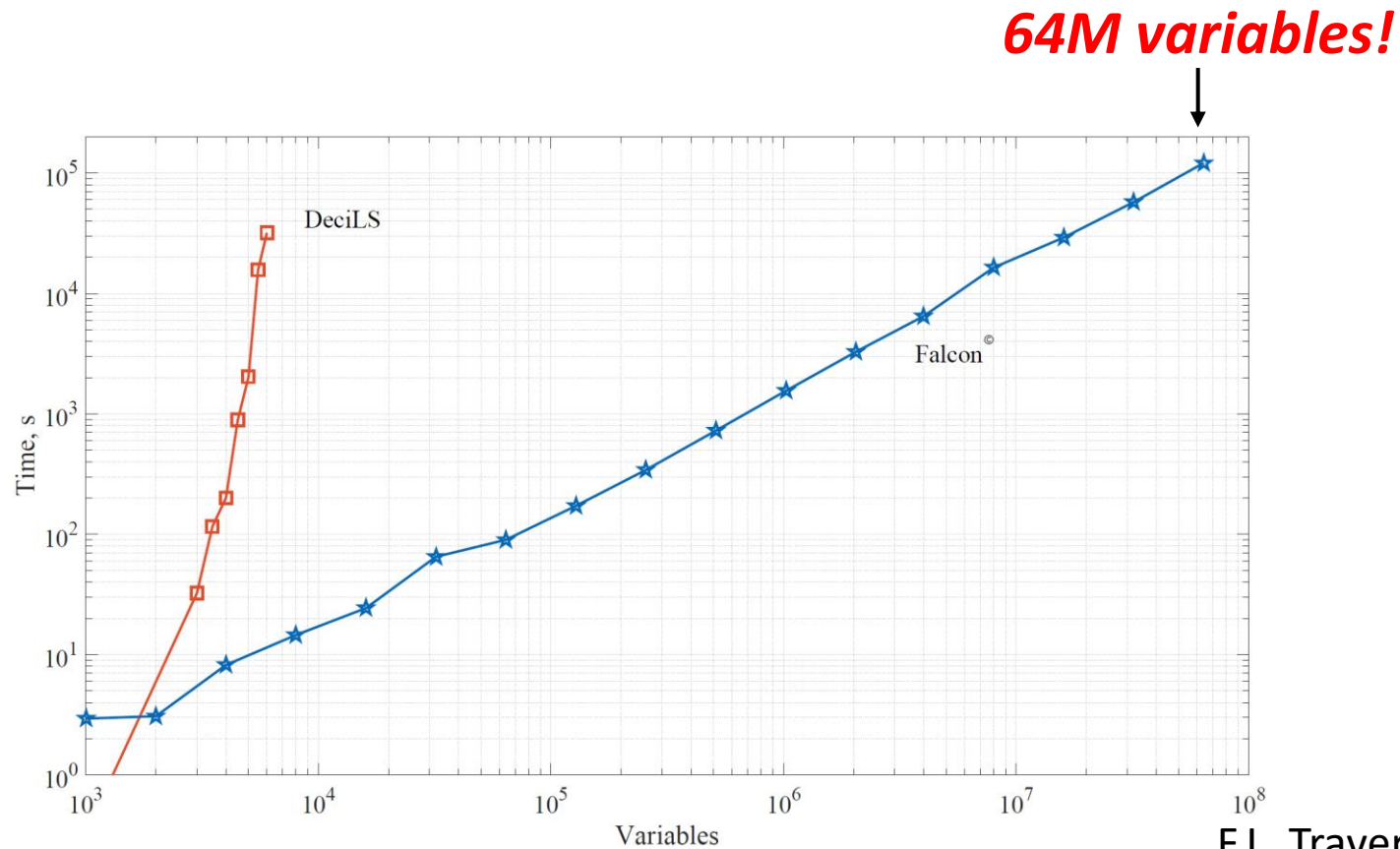
Satisfiability Problem



CNF formula

F.L. Traversa et al., *Complexity* (2018),
F. Shelodon et al., *Arxiv* (2018)

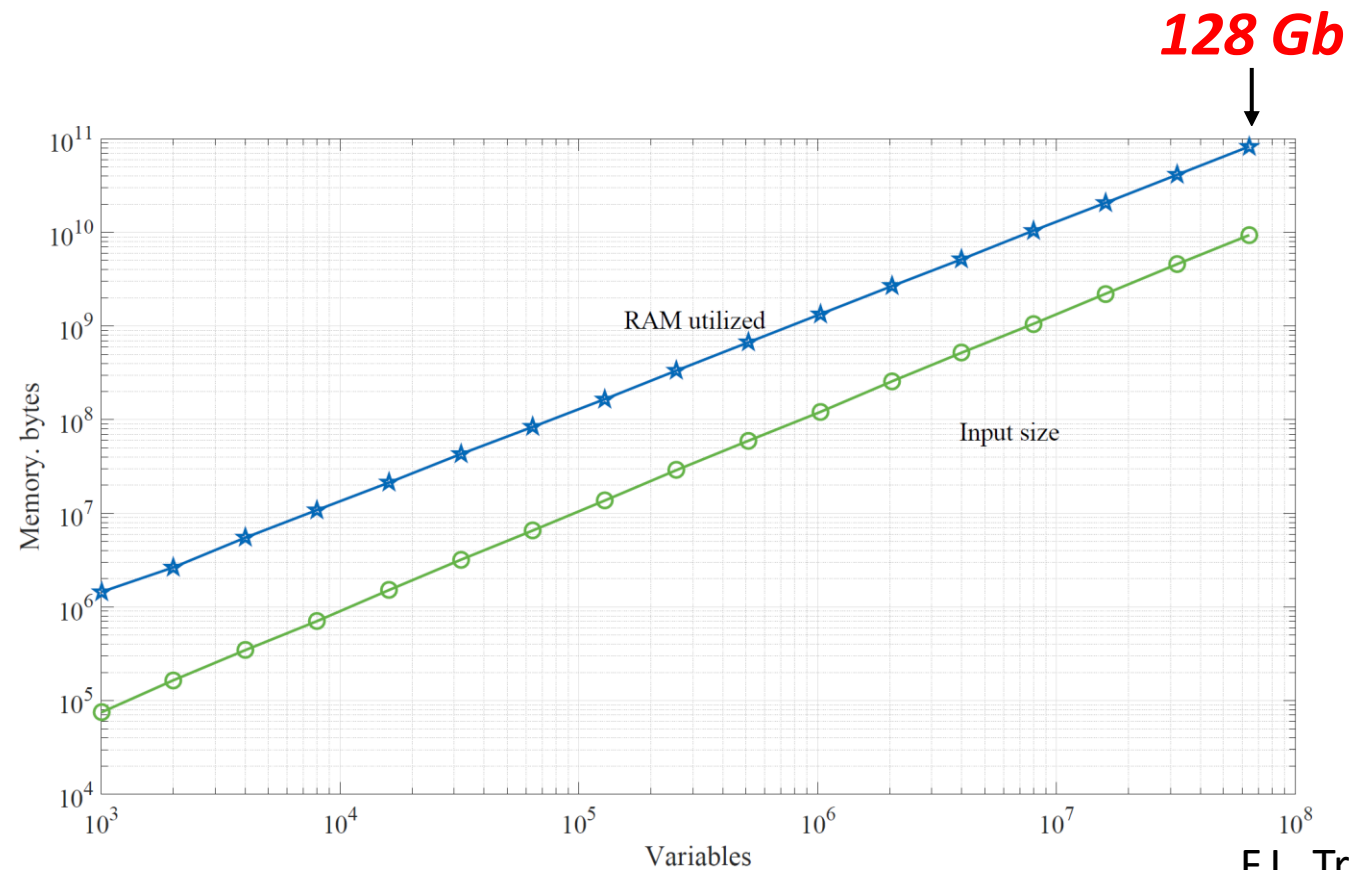
Stress-testing Memcomputing



F.L. Traversa et al., *Complexity* (2018),
F. Shelodon et al., *Arxiv* (2018)

Simulations performed by
Dr. P. Cicotti, NSF San Diego Supercomputer Center
using a MatLab code running on a single Intel Xeon processor

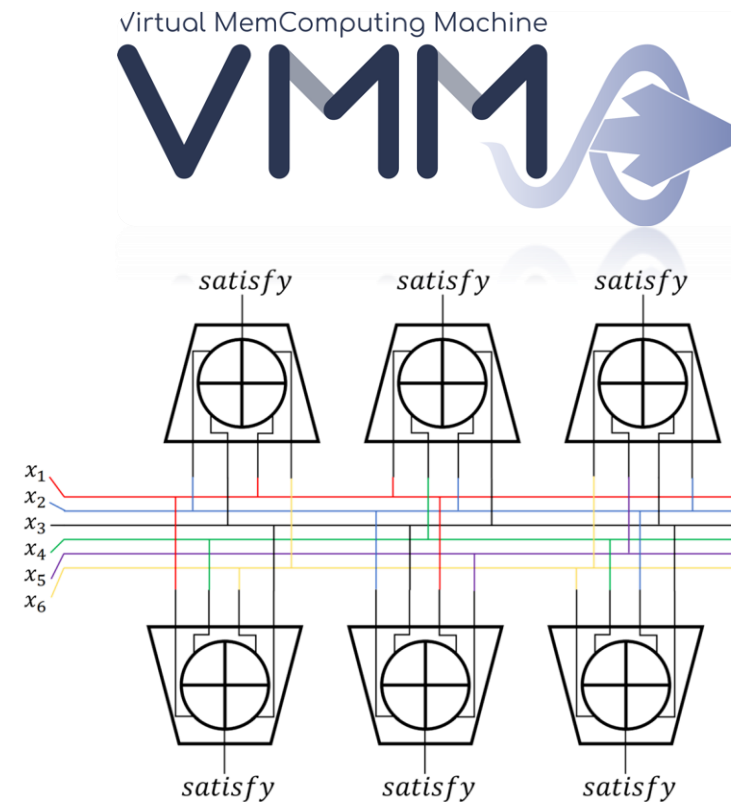
Stress-testing Memcomputing



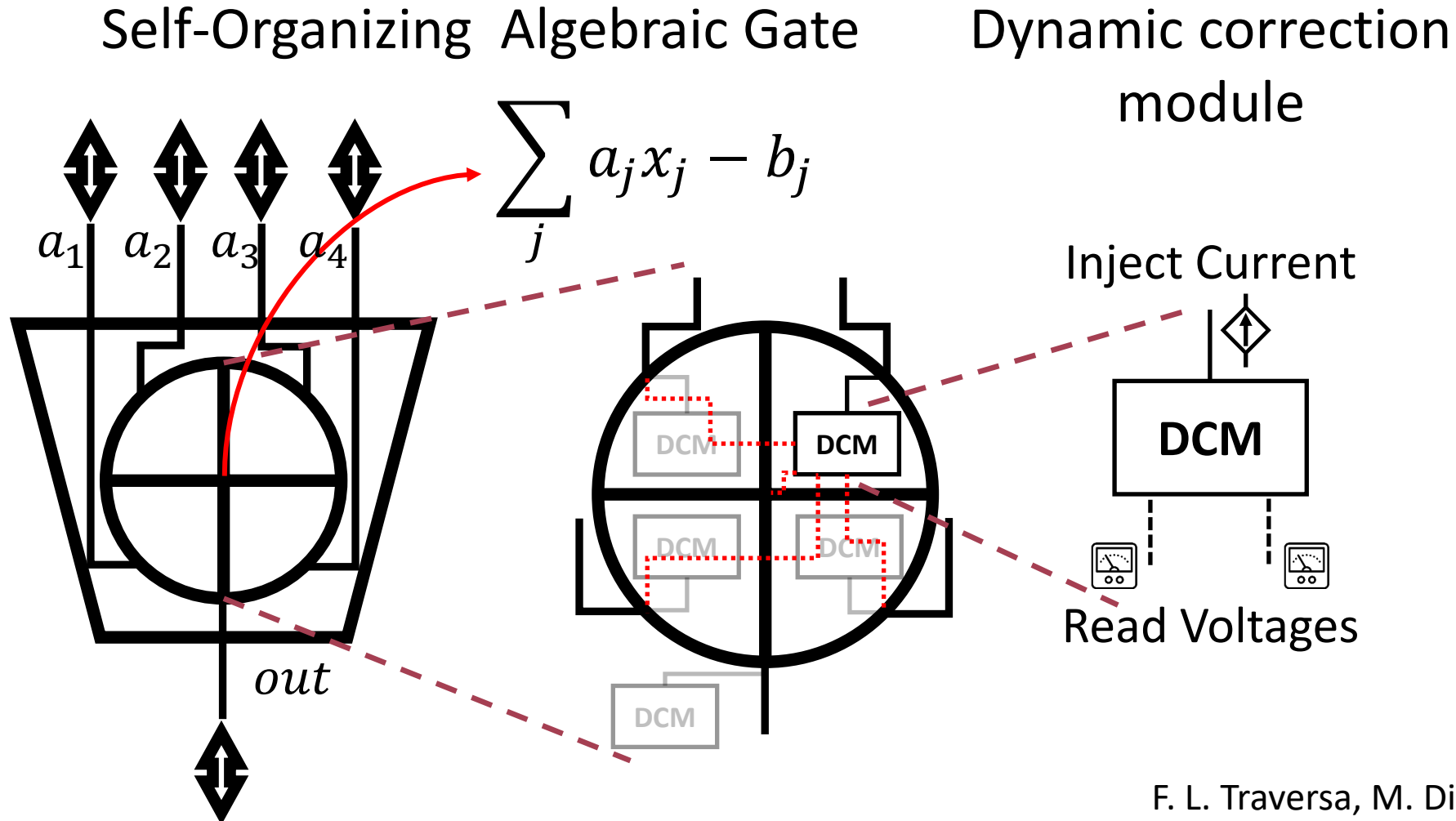
F.L. Traversa et al., *Complexity* (2018),
F. Shelodon et al., *Arxiv* (2018)

Simulations performed by
Dr. P. Cicotti, NSF San Diego Supercomputer Center
using a MatLab code running on a single Intel Xeon processor

Solving Integer Linear Programming Problems



Self-organizing Algebraic Gates



F. L. Traversa, M. Di ventra, *ArXiv* (2018)

GPU & MemCPU

***Distributed architectures are suitable for High
Parallelizable Solutions***

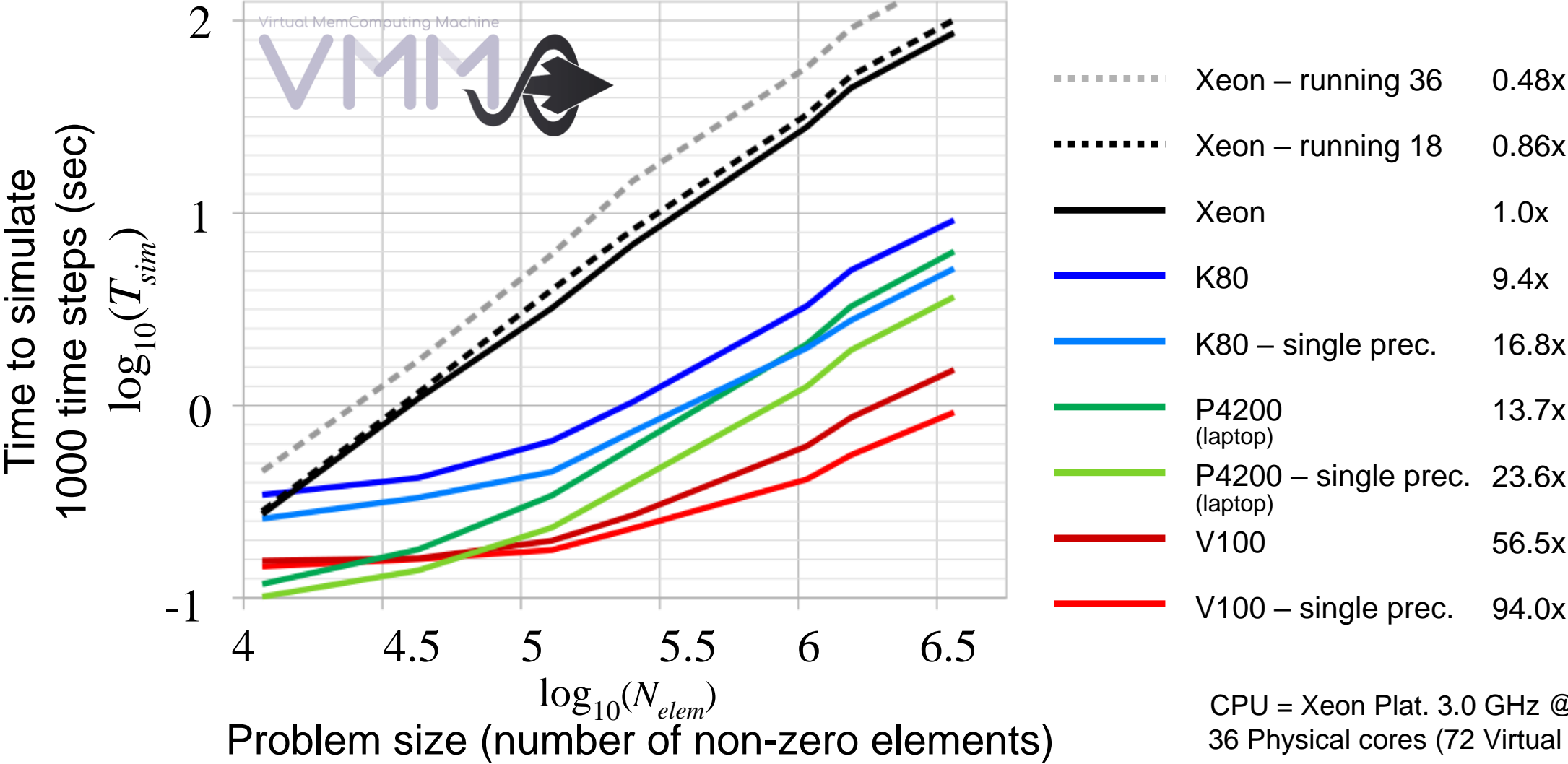
***Cplex, Gurobi, Xpress cannot take advantage of
distributed architectures***

MemCPU can easily run on GPUs

Time to simulate on AWS CPUs and GPUs

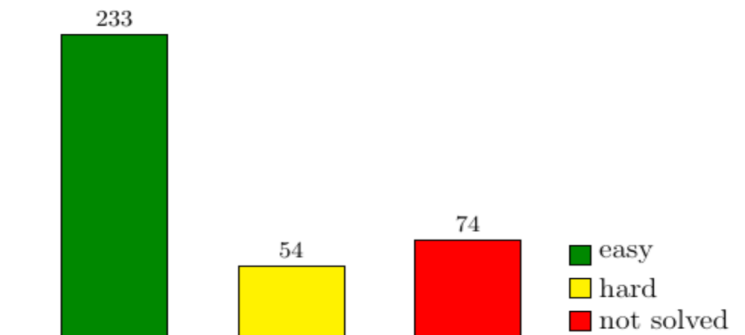
AM LG Problem set

Speed-up of large problems
vs 1 CPU



MIPLIB: Integer Programming

- Released in 2010, MIPLIB2010 is an extremely well known set of instances used to benchmark Mixed Integer Programming (MIP) applications
- Of the ~360 instances in the set:
 - 50+ take at least 1hr to solve using commercial solvers (some still take days to solve)
 - 70+ are still open and unsolved
- Represent difficult problems dealing with:
 - Personnel Scheduling
 - Open Pit Mining
 - Production Lot Sizing
 - Circuit Design
 - Sensor / Telco Equipment Placement
 - Network & Traffic Flow
 - Haplotype Retrieval
 - Protein Folding



MIPLIB Open binary problems

The interesting case of f2000

MIP problem

***f2000 is an
open problem
from MIPLIB-
2010***

SAT problem

***f2000 is an open
from SAT
competition since
2010***

***In 8 years no solver has ever
found a feasible solution***

F. L. Traversa, M. Di ventra, ArXiv (2018)

MIPLIB Open binary problems

The interesting case of f2000

***With MemComputing we have
Found multiple feasible solutions
In a 300 second run***

The first within 60 seconds

MIPLIB Open binary problems

MIPLIB 2017

About ▾

Benchmark

Collection

Download

Help ▾

f2000

binary

variable_bound

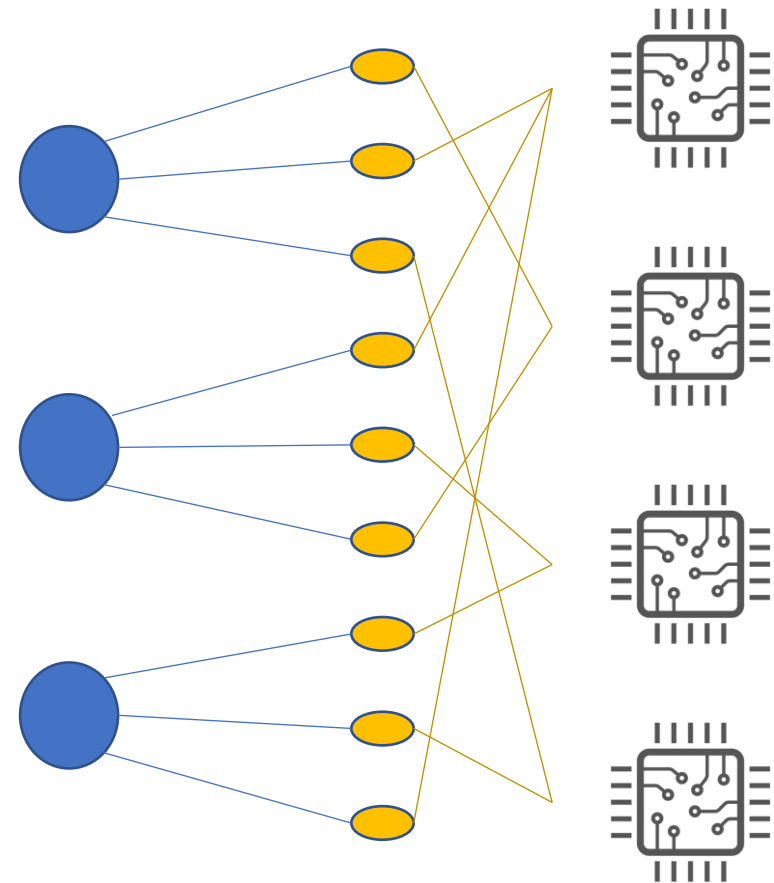
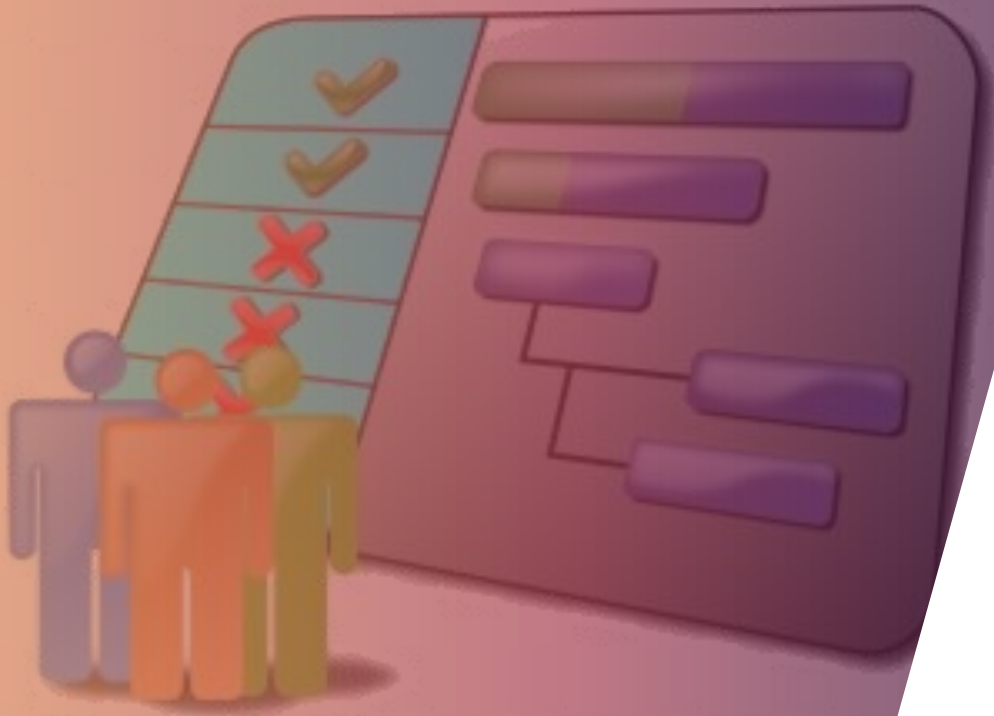
set_covering

Submitter	Variables	Constraints	Density	Status	Group	Objective	MPS File
M. Winkler	4000	10500	7.02381e-04	open	–	1846*	f2000.mps.gz

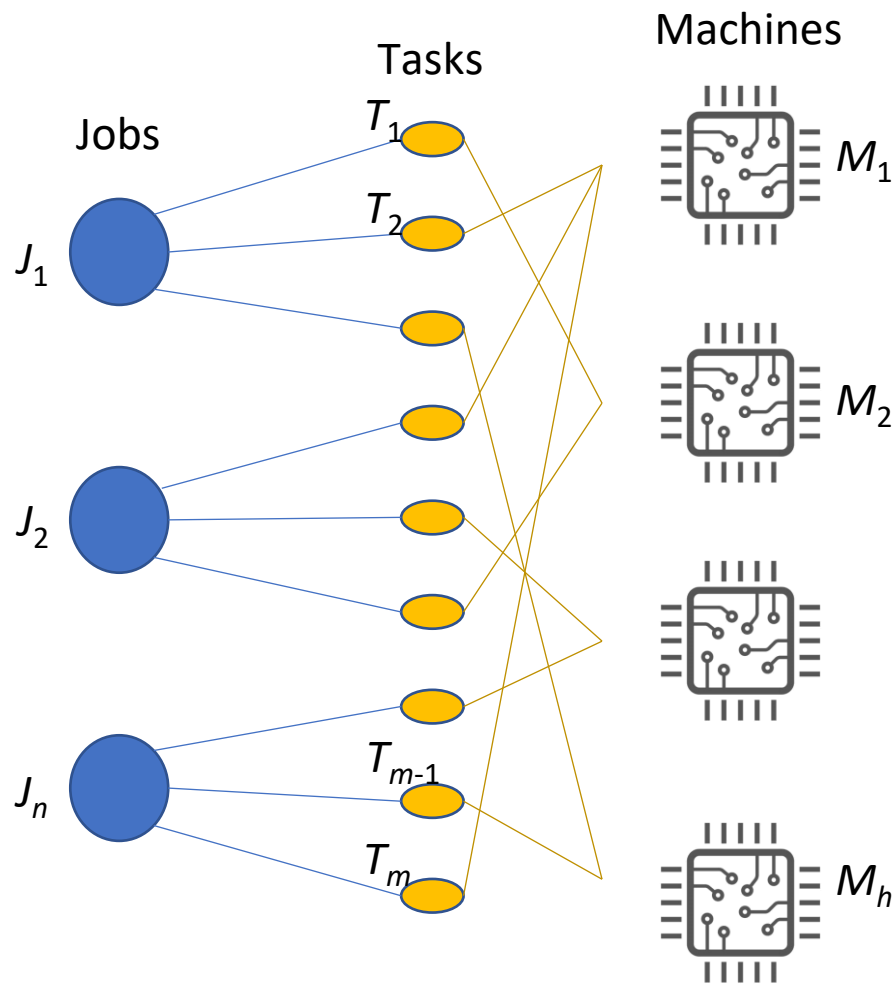
Reformulated SAT instance from the 2010 SAT conference pseudo-Boolean competition

Imported from MIPLIB2010. First feasible solution found by MemComputing, Inc. (<http://memcpu.com/>)

Open-shop scheduling



Problem Definition



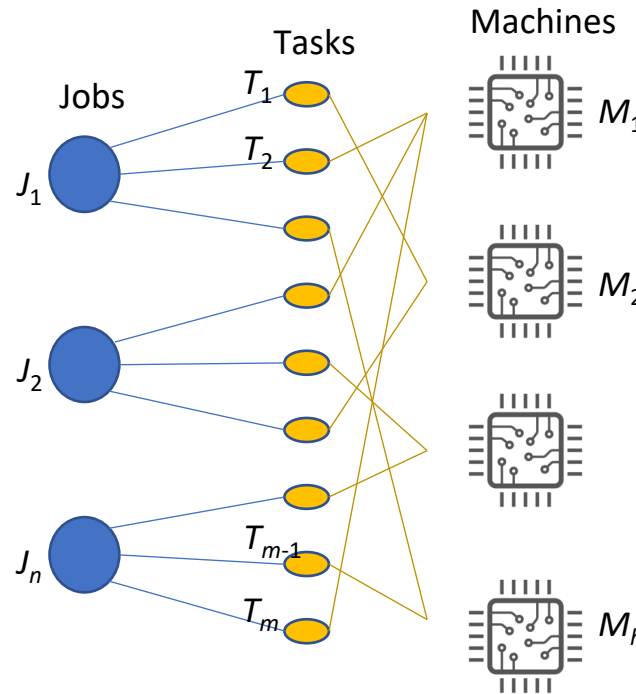
- Each Job has a number of associated tasks that must be run on each of the machines.
- None of the jobs may have tasks running concurrently on a machine, and each machine may only run one task at a time.
- Goal is to find the most efficient schedule for all tasks.

Online vs Offline solution

Online

Schedule the tasks sequentially finding optimal scheduling based only on tasks already scheduled without changing their schedule.

- Run fast
- Low computational complexity
- Compact description (few variables)
- Strong suboptimal scheduling



Offline

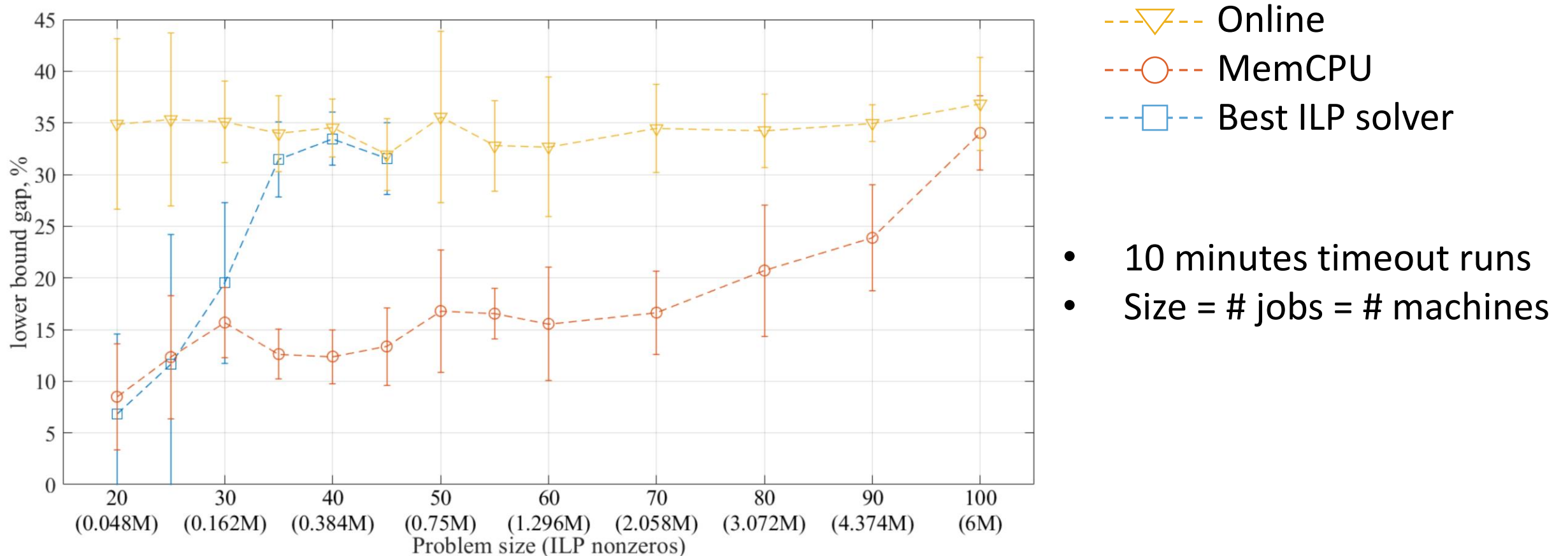
Schedule the all tasks at once finding optimal scheduling.

- Run usually slow
- High computational complexity (NP hard)
- Non compact description (many variables)
- Optimal scheduling

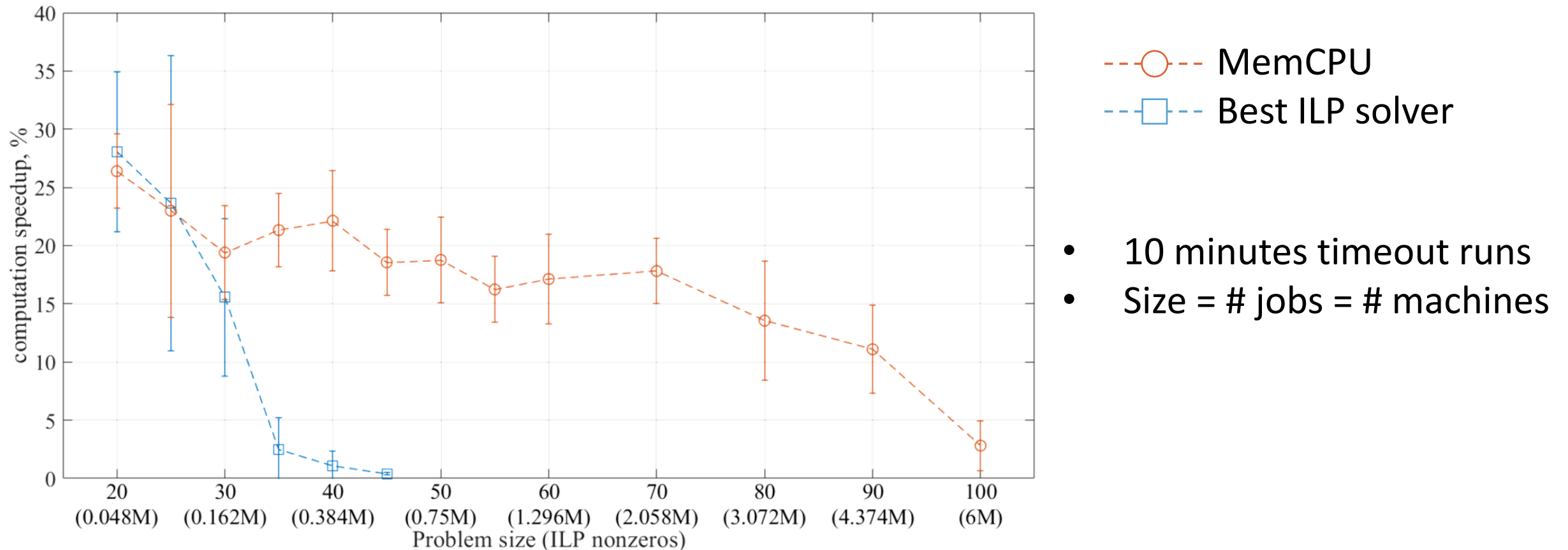
Offline: ILP Formulation

- Given:
 - τ_{jp} : the time to complete j task on p processor
 - T : the upper bound runtime of the schedule
- We have the system of equations describing our ILP as:
 - Minimize: $\max_{jp} \sum_{jp} (t_{jp} + \tau_{jp})$
 - Subject to*
 - $t_{jp} \geq t_{j'p} + \tau_{j'p} - T y_{jj'p}$
 - $t_{jp} + \tau_{jp} \leq t_{j'p} + T(1 - y_{jj'p})$
 - $t_{jp} \geq t_{jp'} + \tau_{jp'} - T y_{jpp'}$
 - $t_{jp} + \tau_{jp} \leq t_{jp'} + T(1 - y_{jpp'})$

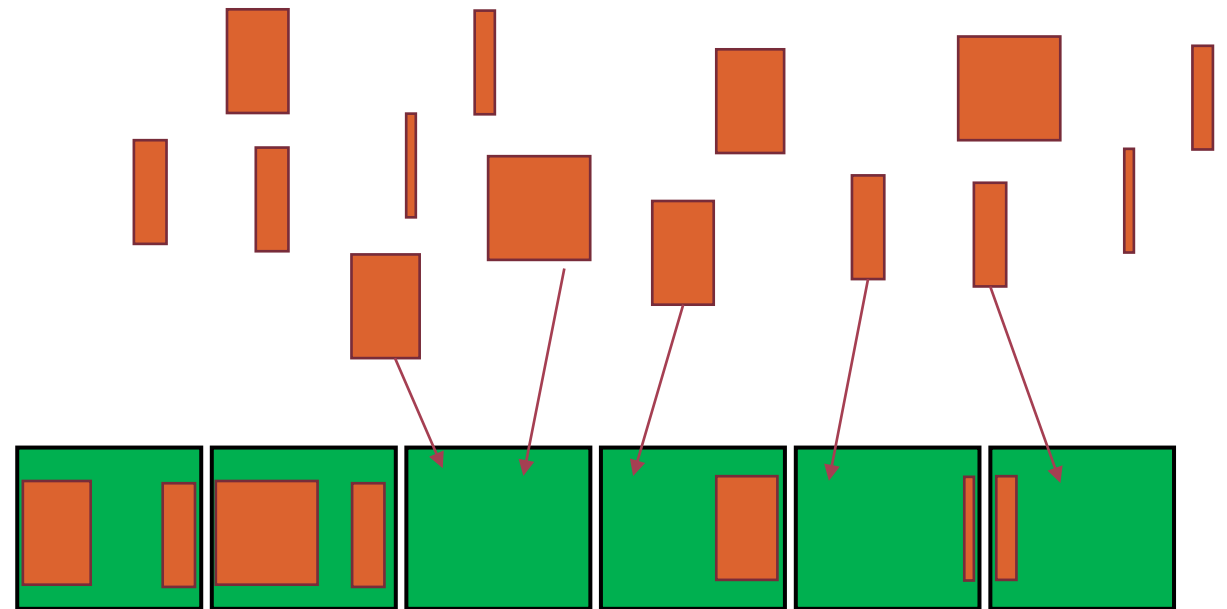
OSS: online vs offline optimized scheduling



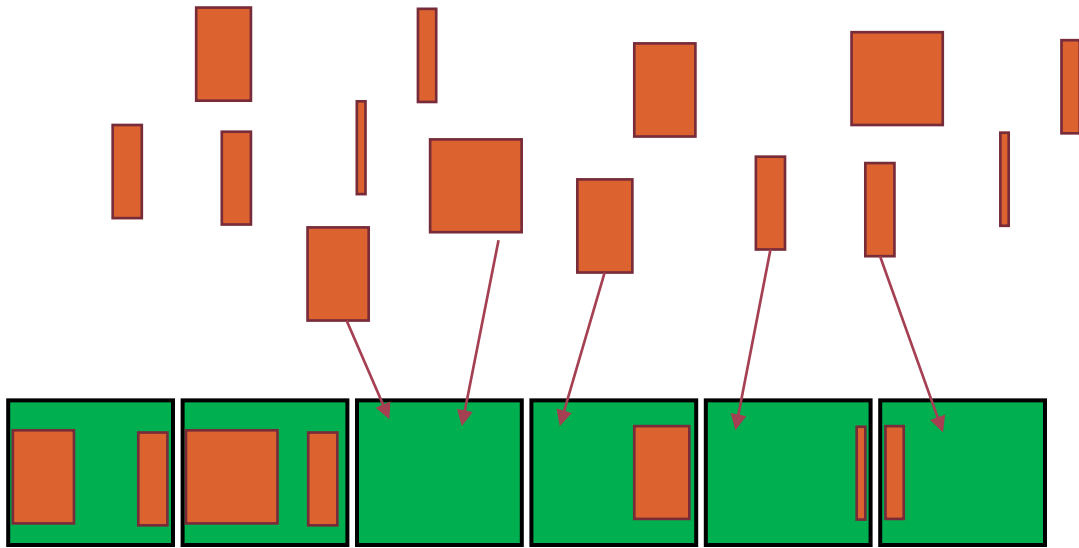
OSS: online vs offline optimized scheduling



Memory allocation (bin packing)



Problem Definition



- A set of M messages of different sizes
- A set of memory banks with capacity B
- Minimize the number of banks to allocate all messages M without exceeding the bank capacity

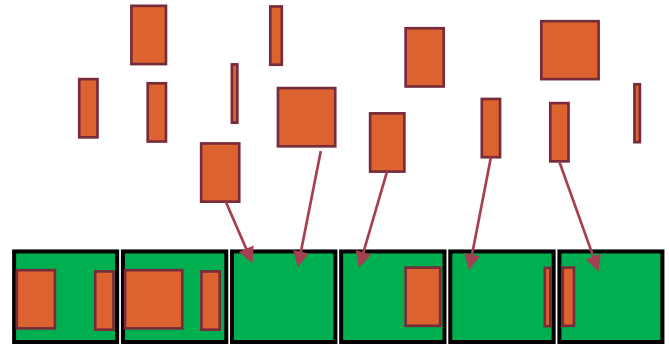
This problem, and its variants, is equivalent to the bin packing problem, a famous NP-hard problem

Online vs Offline solution

Online

Allocate messages sequentially finding optimal allocation based only the current memory allocation without changing it.

- Run fast
- Low computational complexity
- Compact description (few variables)
- Strong suboptimal scheduling, >50% proven suboptimal



Offline

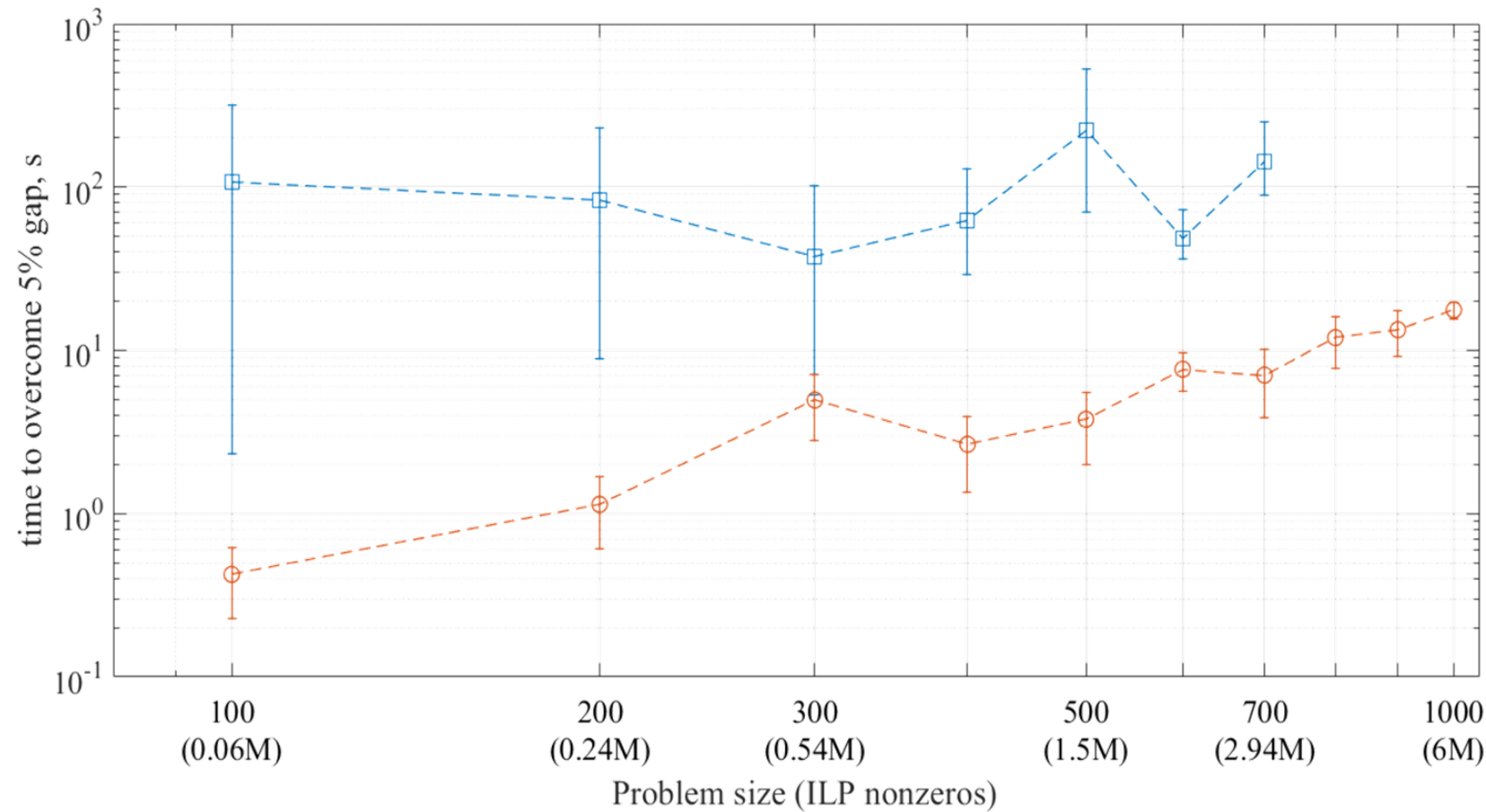
Allocate all messages at once finding optimal allocation.

- Run usually slow
- High computational complexity (NP hard)
- Non compact description (many variables)
- Optimal allocation

Offline: ILP Formulation

- Given:
 - x_{mb} : *binary variable equal to 1 if the message m is in the bin b*
 - y_b : *binary variable equal to 1 if the bank b is occupied*
- We have the system of equations describing our ILP as:
 - Minimize: $\sum_b y_p$
 - Subject to*
 - $\sum_m x_{mb} \leq B y_b$
 - $\sum_b x_{mb} = 1$

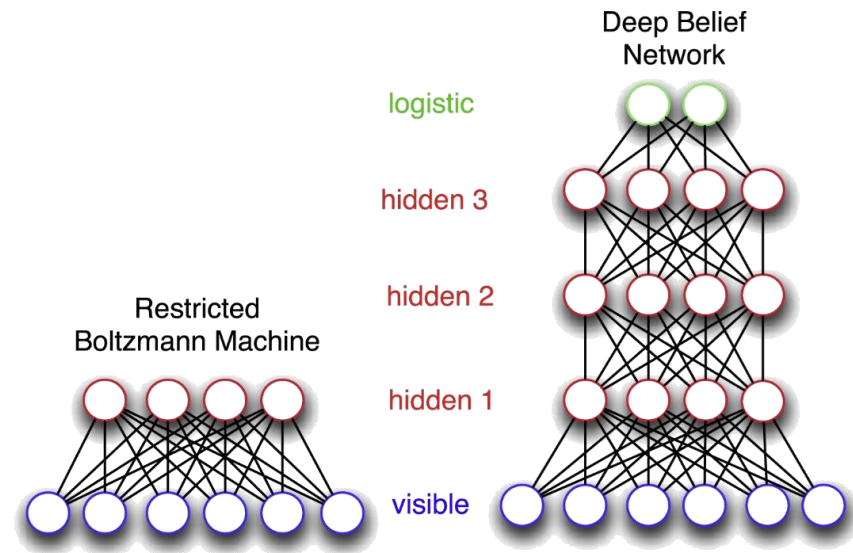
Memory allocation



---○--- MemCPU
---□--- Best ILP solver

- 10 minutes timeout runs
- Size = # message
- Target: find allocation at most 5% above the lower bound.
- Notice, online algorithms are ~50% above the lower bound

Unsupervised Neural Network training



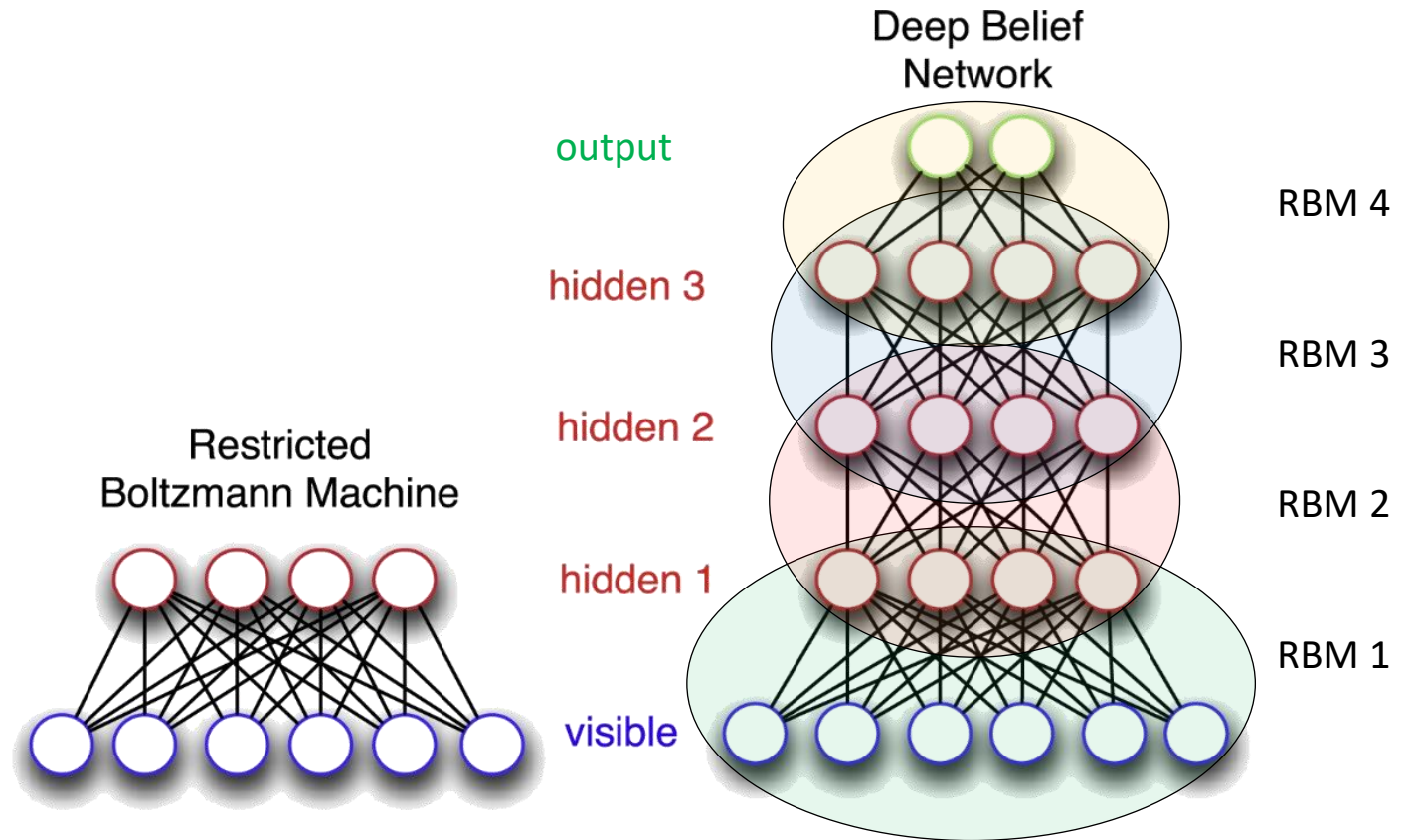
Efficient Restricted Boltzmann Machine training for deep learning

***Pretraining each RBM
(unsupervised learning)***

***Standard method:
Contrastive divergence***

***Training DBN
(supervised learning)***

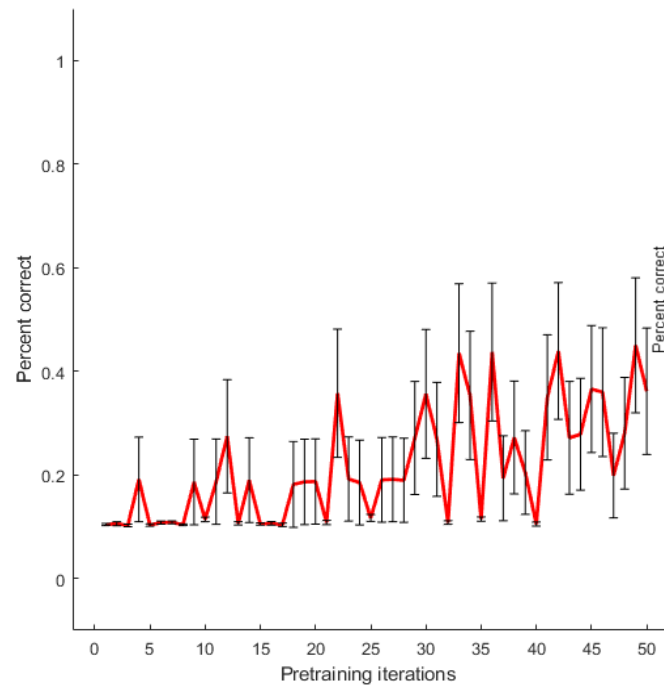
***Standard method:
Backpropagation***



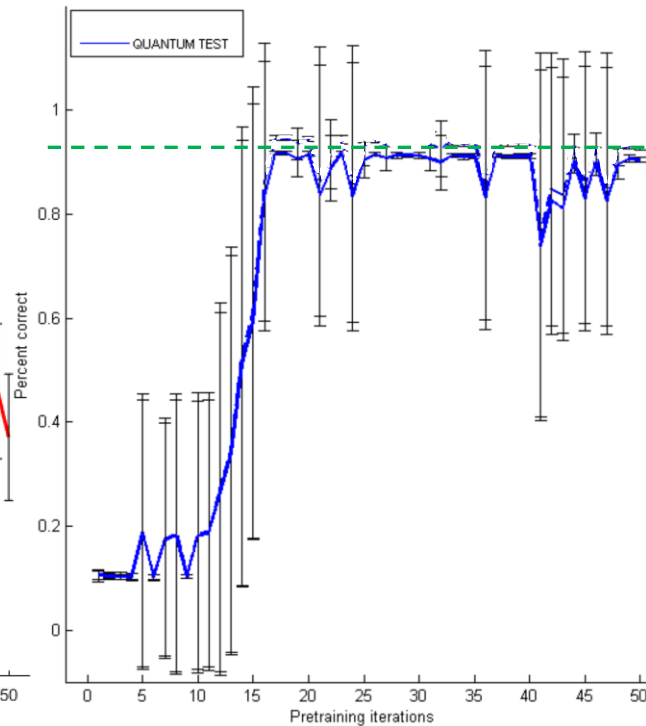
Comparison: after 400 BP iters

Standard Quantum MemComputing

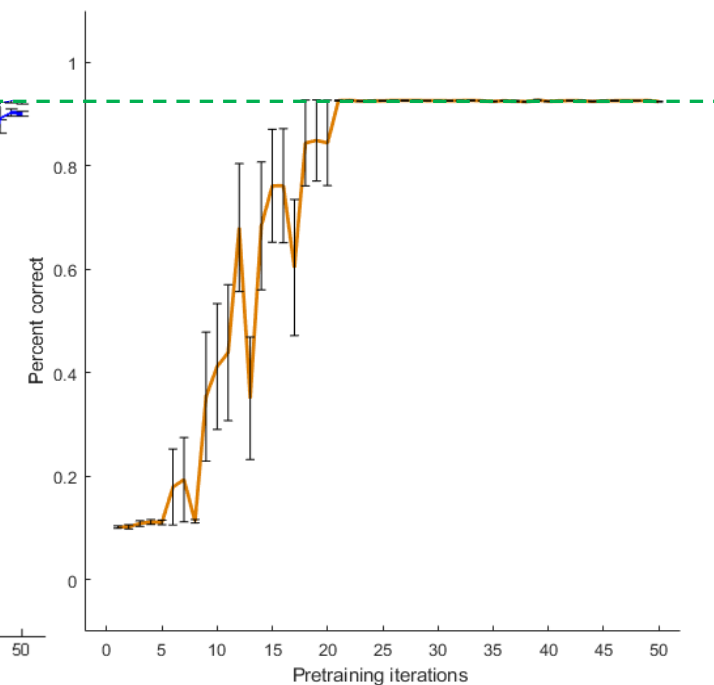
Gibbs




Quantum D-Wave



MemComputing

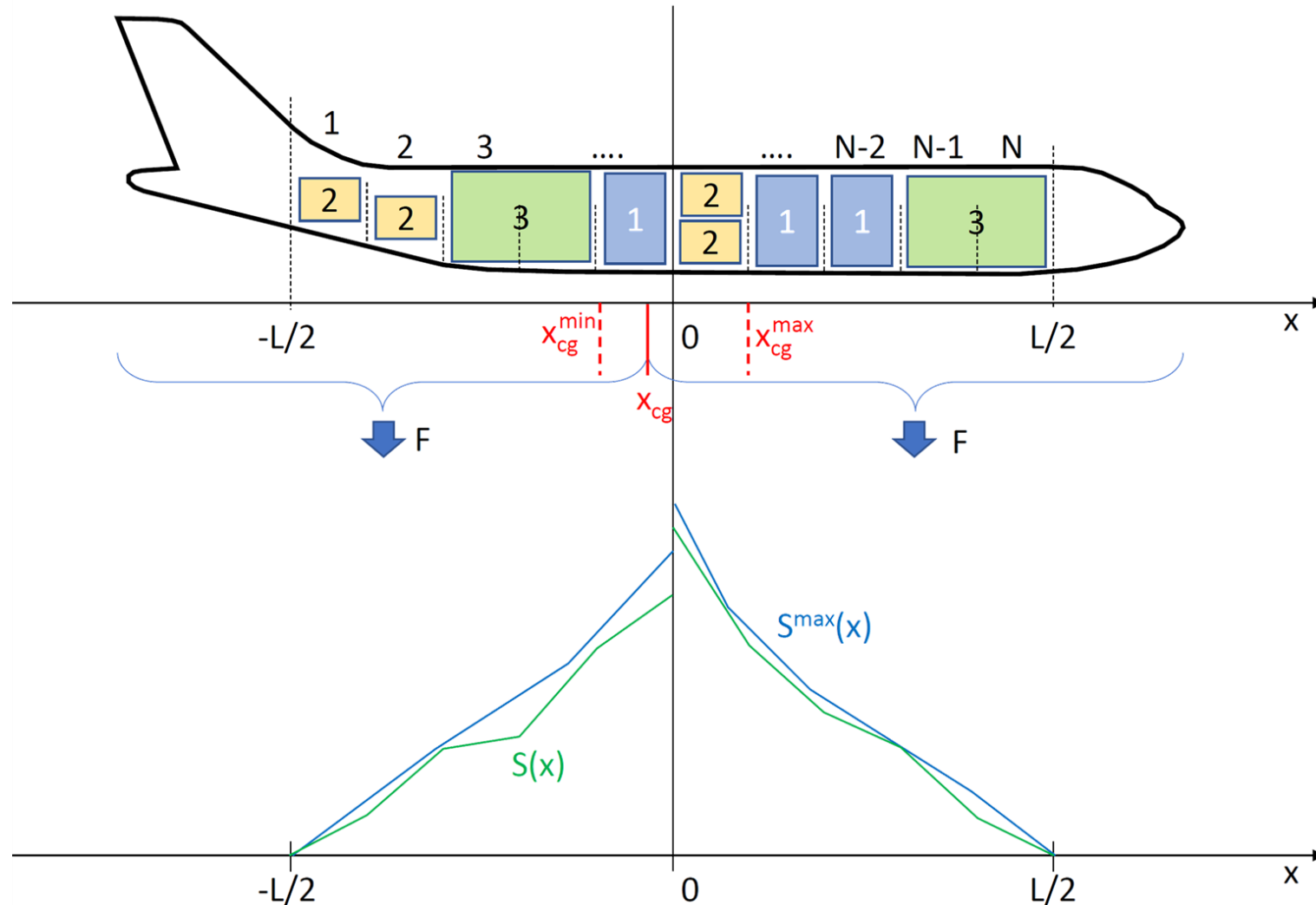


H. Manukian et al., *Neural Networks* (2019)

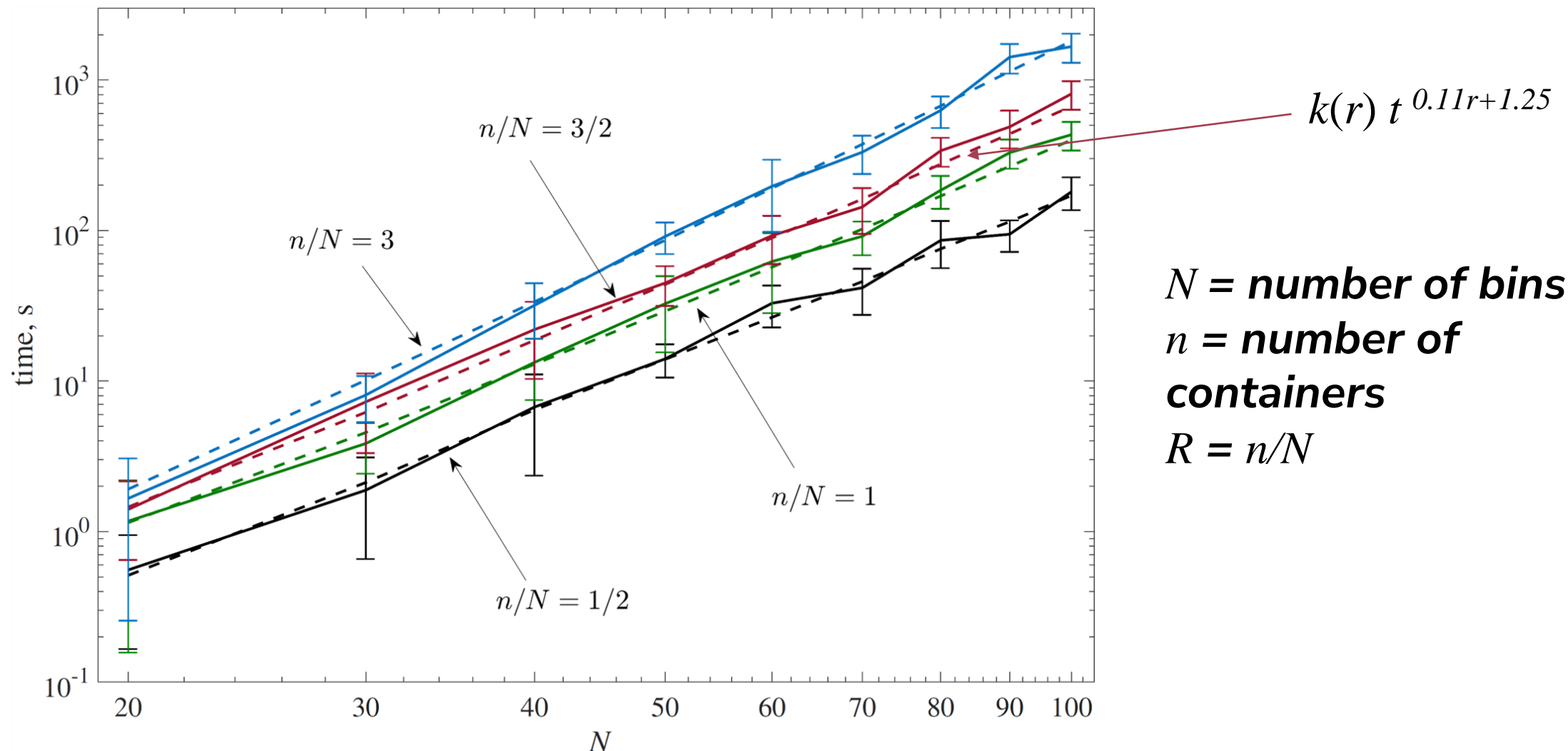
The background of the slide features a dark blue and purple gradient. On the left side, there is a faint, glowing sphere composed of a network of lines and nodes, resembling a complex graph or a molecular structure. Several bright, orange-yellow light rays emanate from the sphere, extending towards the top left corner of the slide.

The 5th Airbus Problem: Aircraft Loading Optimization

Aircraft Loading Optimization



Aircraft Loading Optimization



A photograph of a red and white helicopter on the deck of an offshore oil rig. The helicopter is positioned on a yellow-painted deck area. In the foreground, there are yellow metal railings and stairs. The background shows the ocean and a cloudy sky. The image is partially obscured by a dark blue diagonal overlay on the right side.

Oil & Gas: Helicopter Routing Problem



Helicopter Routing Problem

Goal: Optimize the scheduling/routing of helicopters to offshore rigs

- Represents large operational expense
- Problem is combinatorial in nature
- Must consider:
 - Number of passengers
 - Cargo
 - Helicopter capacity constraints (weight, time, availability)
 - Number of destinations (shore to platform, platform to platform etc.)
- Intractable for today's computers
- Companies rely on heuristic techniques to solve
- Result is sub-optimal operations

Results

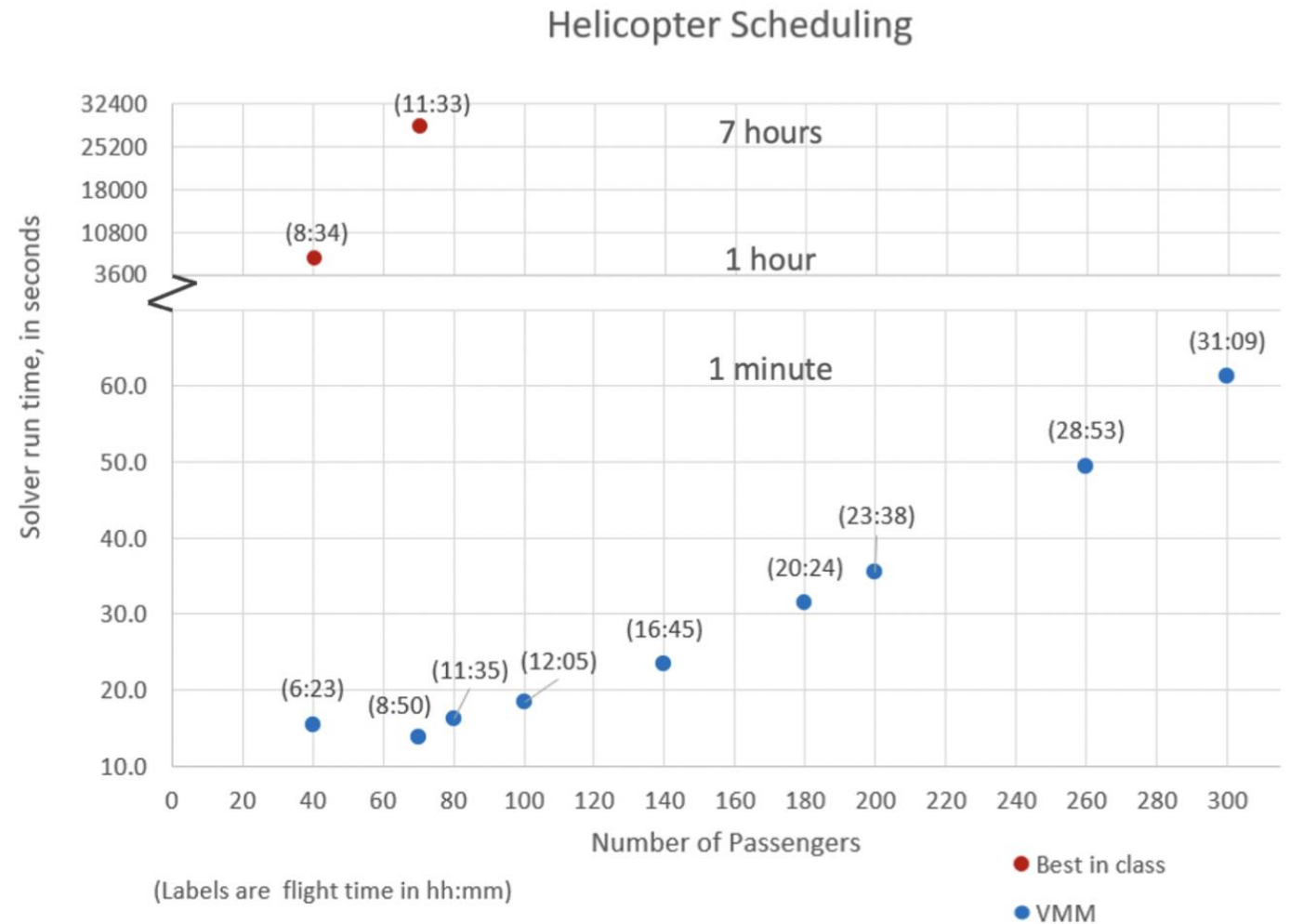
Compared to leading commercial solver

- Commercial solver takes hours for small instances
- Commercial solver unable to scale past 80 passengers
- VMM scales polynomially
- VMM finds near optimal solutions at scale in seconds
- Ability to improve operations & bottom line

Virtual MemComputing Machine



Scheduling for:
25 rigs, 2 kinds of helos, 1 heliport, varying # of passengers



THANK YOU

WWW.MEMCPU.COM



FABIO L. TRAVERSA



ftraversa@memcpu.com